ARO 21065...

2

SOUND PROPAGATION THROUGH A TURBULENT ATMOSPHERE: EXPERIMENTAL TECHNIQUES AND DATA ANALYSIS

Final Report

Henry E. Bass, Lee N. Bolen, and John Noble

THE UNIVERSITY OF MISSISSIPPI
PHYSICAL ACOUSTICS RESEARCH GROUP
DEPARTMENT OF PHYSICS AND ASTRONOMY

2

SOUND PROPAGATION THROUGH A TURBULENT ATMOSPHERE: EXPERIMENTAL TECHNIQUES AND DATA ANALYSIS

Final Report

Henry E. Bass, Lee N. Bolen, and John Noble

PARGUM Report #87-02

CS

19 October 1987
U.S. Army Research Office
Contract #DAAG-29-84-0013
Physical Acoustics Research Laboratory
The University of Mississippi
University, Mississippi 38677

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Physical Acoustics Research Lab Univeristy of Mississippi 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION U. S. Army Research Office
6c. ADDRESS (City, State, and ZIP Code) University, MS 38677	7b. ADDRESS (City, State, and ZIP Code) P. O. Box 12211 Research Triangle Park, NC 27709-2211
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U. S. Army Research Office 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAG29-84-K-0013
8c. ADDRESS (City, State, and ZIP Code) P. O. Box 12211 Research Triangle Park, NC 27709-2211	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Sound Propagation Through a Turbulent Atmosphere: Experimental Techniques and Data Analysis (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)
Henry E. Bass, Lee N. Bolen, and John Noble

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 12/1/83 TO 7/31/87	871019	192

16. SUPPLEMENTARY NOTATION The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Sound propagation, turbulence, outdoor sound
cont p 1			

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This is the first of a series of reports on a three-year study of sound propagation through a turbulent atmosphere. This report documents the experimental configuration and describes data analysis. The data analysis includes plots of the real and imaginary parts of the acoustic pressure as a function of time (scatter plots), probability of observing a particular amplitude, and the more familiar structure functions.

A preliminary analysis of data suggests reasonable agreement in structure functions at frequencies of 500 Hz and above. At lower frequencies, phase and log amplitude structure functions are both larger than predicted from theory. A tentative explanation for this difference is under development and will be presented in the third of the three volume series. The second volume will be devoted to refractive effects.

Outdoor

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Table of Contents

SOUND PROPAGATION THROUGH A TURBULENT ATMOSPHERE:
EXPERIMENTAL TECHNIQUES AND DATA ANALYSIS

Final Report

Henry E. Bass, Lee N. Bolen, and John Noble

PARGUM Report #87-02

Abstract

This is the first of a series of reports on a three year study of sound propagation through a turbulent atmosphere. This report documents the experimental configuration and describes data analysis. The data analysis includes plots of the real and imaginary parts of the acoustic pressure as a function of time (scatter plots), probability of observing a particular amplitude, and the more familiar structure functions.

A preliminary analysis of data suggest reasonable agreement in structure functions at frequencies of 500 Hz and above. At lower frequencies, phase and log amplitude structure functions are both larger than predicted from theory. A tentative explanation for this difference is under development and will be presented in the third of the three volume series. The second volume will be devoted to refractive effects.

Sound Propagation Through a Turbulent Atmosphere:
Experimental Techniques and Data Analysis

I. Introduction

The propagation of sound waves close to the ground is a complex problem involving many interesting mechanisms. In addition to geometrical spreading and molecular absorption, which are reasonably well understood, the three main mechanisms which influence the acoustic field are reflection with phase change due to the finite impedance of the ground, refraction by wind and temperature gradients, and scattering by atmospheric turbulence. Outdoor sound propagation in a turbulent medium is not a well understood process and has only recently begun to receive serious attention. This report will present experimental data taken under several different meteorological conditions including various degrees of turbulence and different geometrical configurations. This report also contains the results of a preliminary analysis of the data.

The first section will describe the experimental configuration of the experiments which were undertaken over a period of two years. A complete set of geometrical configurations will be presented along with the analysis procedure for processing the acoustical data. A unique technique for examining the fluctuations in acoustic amplitude is described in this section. This analysis was completed for each microphone in the array over a range of frequencies starting at 62.5 Hz and going up to 8000 Hz. The next section describes the meteorological information that is available for the experiments. Temperature and wind velocity vary strongly with height within the first few meters above the ground. Because of this, the wind speed, wind direction, and temperature measurements are given as a function of height. Sound speed profiles have been computed for each of the runs where weather information is available. The third section deals with extracting sound pressure levels from the analyses described in Section II.

The fourth item discussed will be the form of the complex amplitude of the sound wave as it propagates through a turbulent medium. As an acoustic wave propagates a distance r through the atmosphere from a point source S to a receiver R, atmospheric turbulence causes fluctuations in

the acoustical refractive index, the effect of which is to produce fluctuations in the phase and amplitude of the sound field at the receiver. In order to observe these fluctuations, a set of plots, referred to as scatter plots, are made which show the fluctuation of the complex amplitude over time. These plots will provide clues about the interaction between the acoustic wave and the turbulence. From these scatter plots, an empirical argument emerges which provides a statistical distribution having the same characteristics as the data. A comparison between the statistical distribution and the data is also presented.

Finally, the phase and log-amplitude structure functions are calculated for the data runs. The calculations are compared with the structure functions determined by Daigle.

II. Experimental Configuration

Measurements were made over a relatively flat open farm land over a period between mid-June 1984 and mid-July 1985. The sound source was driven by a tape which had, pre-recorded, a signal consisting of a mixture of seven pure tones centered at one octave spacing beginning with 62.5 Hz. A run consisted of an eight minute record of signals received simultaneously at five microphones mounted one meter above the ground surface and one microphone mounted near the source. The received signals were recorded on a seven channel TEAC R-81 tape recorder. Typically, at least one run was made in the morning and one run in the afternoon. This allowed for propagation through different turbulence conditions. During a typical run, the first five channels were used to record the signals from the array microphones, the sixth was used to record the reference microphone and channel seven was used for a voice log.

The measurements were made using four different types of geometries. The most common geometry had the source on the ground with a transverse array of five microphones one meter above the ground about 100 meters from the source and one microphone five meters from the source also one meter above the surface. Another common geometry had the source on top of a 100 foot tower with a transverse array of five microphones on the ground with one microphone about five meters from the source at the same height as the source. One experiment, December 13

- Run 3.1, was performed with the source on top of the tower and a vertical array of microphones ranging from 0 meters to 2.57 meters above the ground. Another run, December 13 - Run 2.1, was conducted with the source on top of the tower with a longitudinal array of microphones on the ground. A complete list of geometries for all of the runs is found in Appendix A.

The transverse array was chosen so that the phase and log-amplitude structure functions could be calculated for various transverse separation distances. This gives a measure of the fluctuations in the sound field after propagating some distance along almost parallel paths.

One quantity of interest is the fluctuation in the amplitude. In order to observe these fluctuations, we devised the analysis procedure shown in Figure 2.1. The analysis begins by retrieving the data from tape and passing it through a 1/3 octave filter set at one of the broadcast frequencies. The output from the filter was sent to a multichannel analyzer (MCA) which stored the amplitude of each acoustic half cycle in a channel number proportional to the amplitude. The number of counts in each channel is proportional to the probability that an acoustic half cycle will have a particular amplitude. The MCA accumulated data over the time of the run. After the accumulation was completed, the MCA showed amplitude distribution. After examining these for various frequencies, we found that the amplitude distributions were of two basic shapes; a Gaussian or Rayleigh as shown in Figure 2.2 and Figure 2.3. From these distributions, the relative sound pressure levels were calculated by taking from the distribution plots the difference between the mode of the array microphone and the mode of the reference microphone. The procedure for calculating the relative sound pressure levels is outlined in detail in Chapter IV. Later, the amplitude distribution plots will be used to fit a statistical distribution.

The next analyses yield the phase and log-amplitude structure functions and scatter plots. The configuration of the analysis equipment is given in Figure 2.4. The output from each channel of the recorder (other than seven) was passed through a 1/3 octave filter set at one of the broadcast frequencies to a multichannel ADF12F A/D converter connected to a Masscomp 5535 mini-main frame computer. The data was stored in a 16 bit binary file format for further analysis. The digitization rate was twelve times the signal frequency.

AMPLITUDE DISTRIBUTION
APPRARATUS SETUP

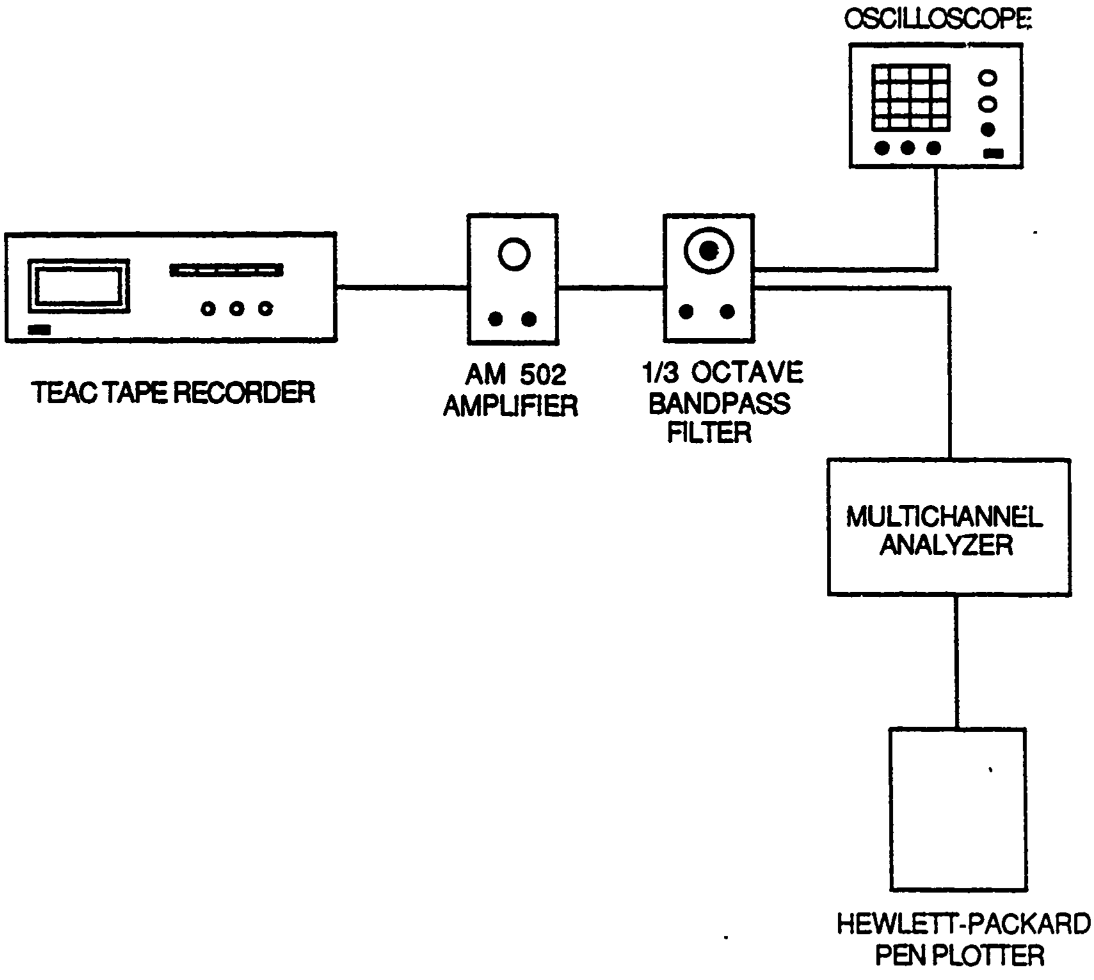


Figure 2.1

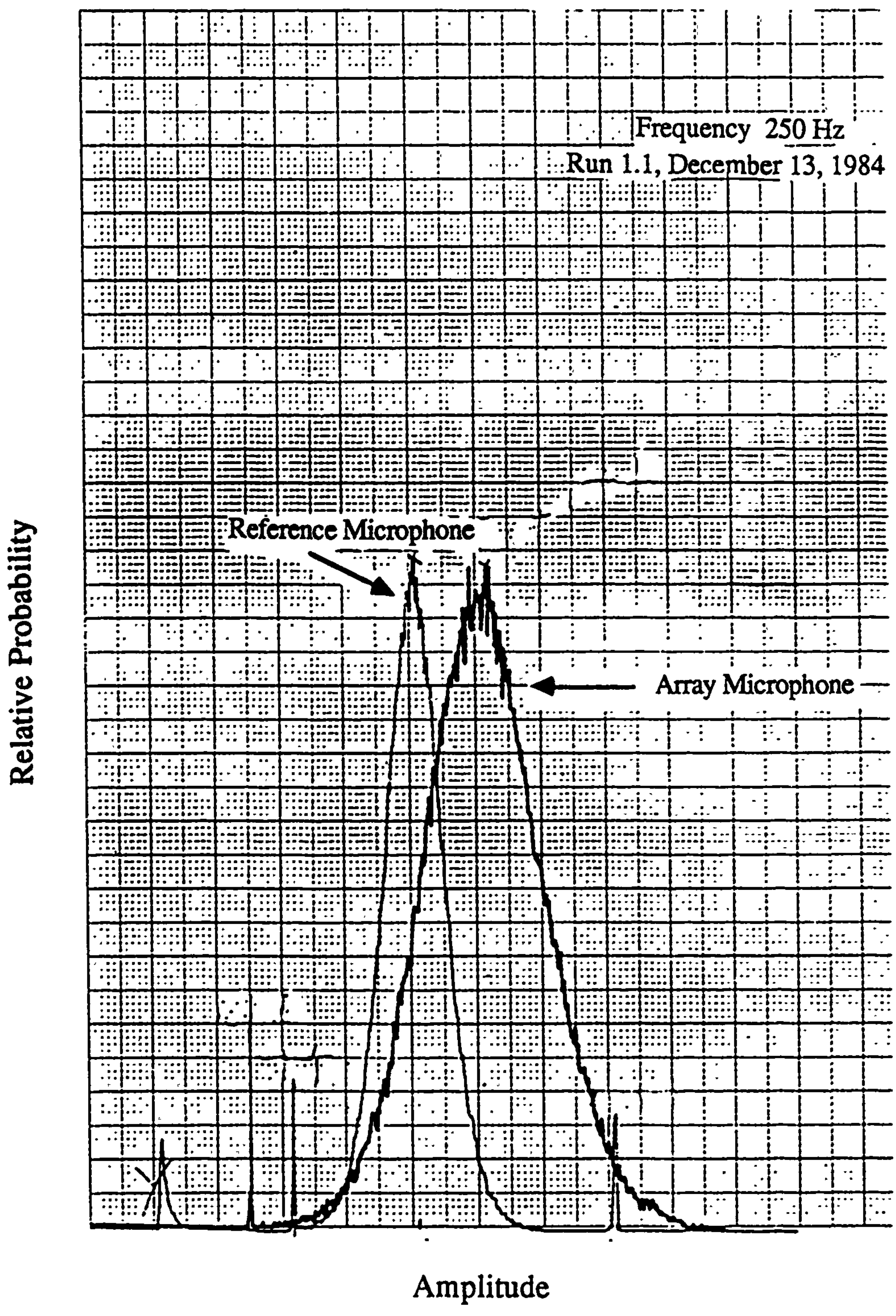


Figure 2.2 Amplitude Probability from MCA

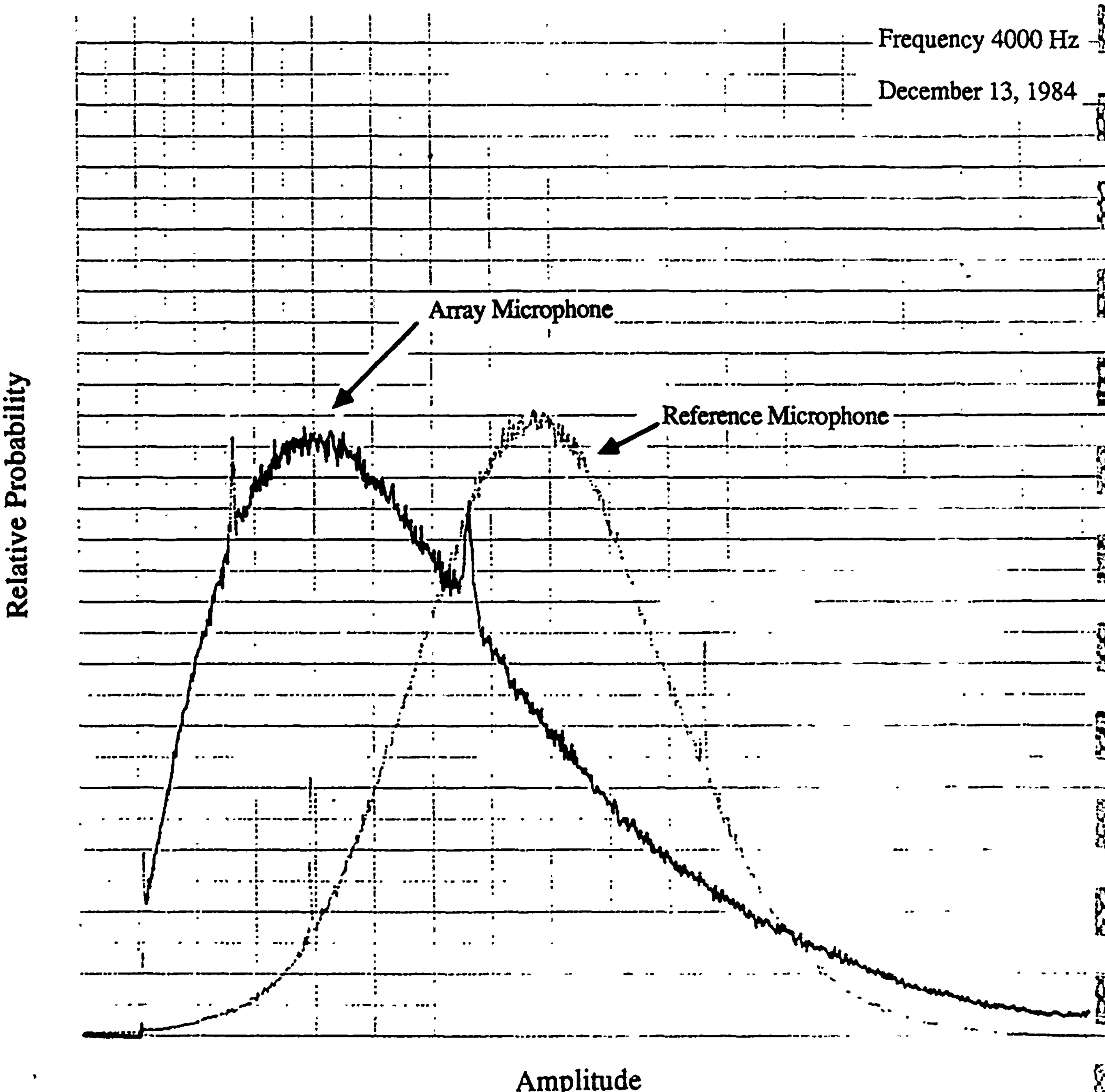
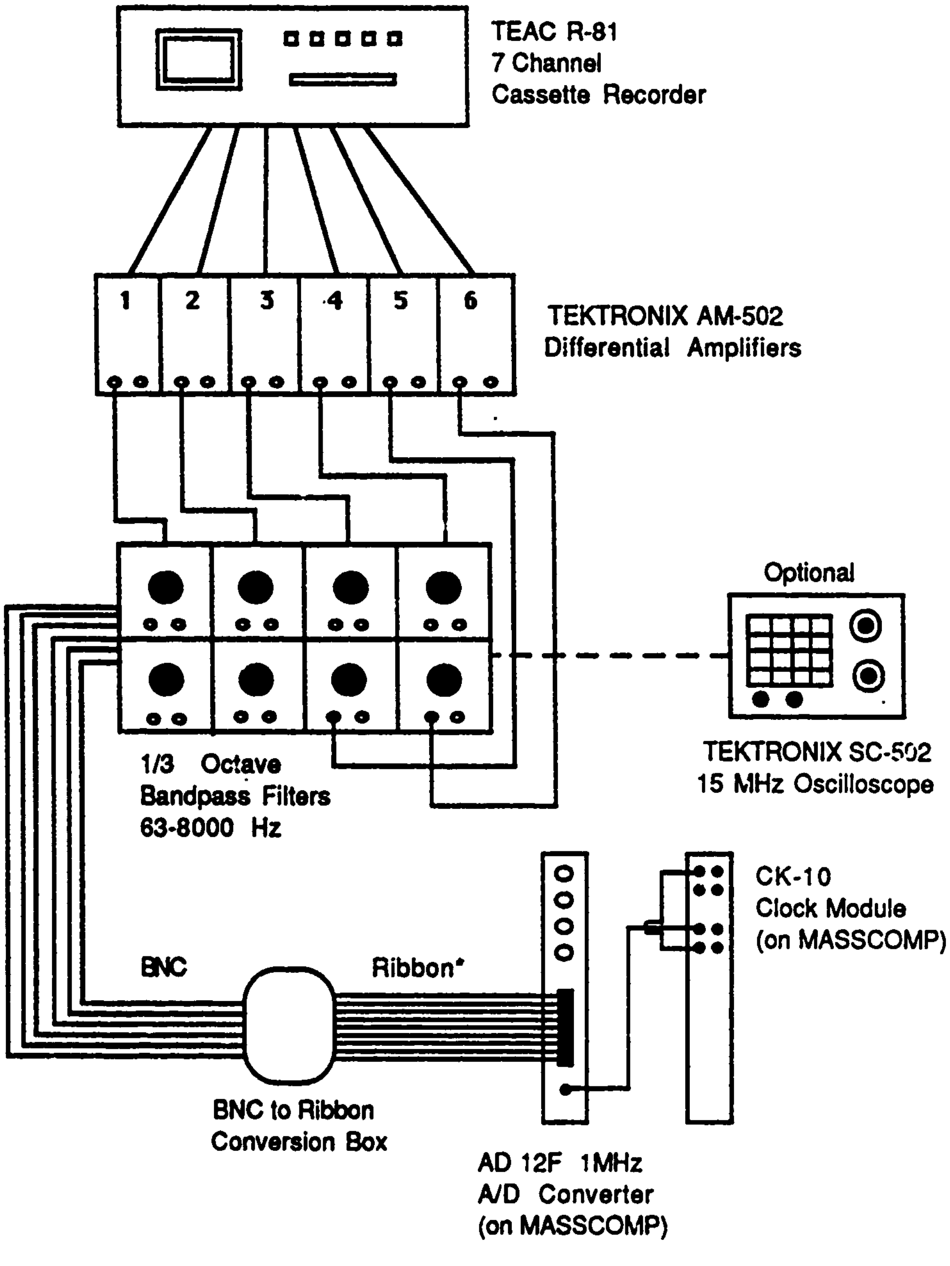


Figure 2.3 Amplitude Probability from MCA

TURBULENCE DATA ANALYSIS APPARATUS SETUP



* SPETRA-STRIP (Gray with red edge, 28 AWG, 34 Conductor Cable)

Figure 2.4

The program phaz.f in Appendix B is used to calculate the phase and amplitude difference between any two channels for each acoustic half cycle. It accomplishes this by reading in 1024 points from each channel at one time and subtracting the dc bias from each channel. Next, the program searches through the data and locates the sign changes on each channel. The time of zero crossing is then estimated. A similar procedure based on change of slope gives the corresponding amplitude located between the crossings. After the time for each zero crossing has been calculated, the difference between this time and the corresponding time on another channel is calculated which is related to the phase difference between the two channels. The ratio of amplitudes for the two channels being compared is also computed for each half cycle. This process continues until all possible comparisons between channels have completed. The output from this program is used by the scatter plot and structure function routines described later.

III. Weather Profiles

Meteorological effects can have a significant effect on the received sound field. Large eddies are formed in the atmosphere by instabilities in the thermal and viscous boundary layers at the surface of the ground. Futher instabilities cause these eddies to break down progressively into smaller and smaller sizes until the energy is finally dissipated by viscosity in eddies approximately 1 mm in size. A statistical distribution of eddies, which we call turbulence, is therefore present in the atmosphere at all times. The intensity of the turbulence, however, is strongly dependent on meteorological conditions.

While the effects of wind and temperature gradients appear similar in the graphs in Appendix C, the following differences should be noted. Because temperature is a scalar quantity, the refraction of sound produced by lapse or inversion conditions is the same in all horizontal directions. Wind, however, produces refraction which is nonuniform in direction according to the vector component of wind relative to the direction of propagation. Thus, the refraction produced

by wind is zero when the sound propagates directly crosswind, and increases progressively as the direction of propagation deviates from this condition.

Temperature and wind velocity vary strongly with height within the first few meters above the ground. These vertical gradients produce steep sound speed profiles close to the ground. In addition to the vertical gradients, the temperature and wind velocity fluctuate about their mean. The resulting random fluctuations of the refractive index scatter sound, which leads to random fluctuations in the phase and amplitude of the sound wave. Even when the turbulence is sufficiently weak that it has negligible effect on the sound field in free space it is still sufficient to affect the sound field above the boundary, especially in regions of destructive interference where the sound level is critically dependent on phase relationships.

For most of the experiment measurements of the wind velocity and temperature were made at four heights simultaneously. These points were located at 3, 10, 30, and 110 feet. An example of the output from the sensors is given in Table 3.1.

Time	Height	Wind Speed (m/s)			Wind Direction			Temperature (C)		
	feet	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min
11:25	110	8.0	11.0	6.1	34.8	404.3	386.4	2.7	3.1	2.3
	30	6.2	7.8	4.7	52.3	423.3	402.2	2.9	3.3	2.3
	10	5.8	7.3	4.1	226.5	248.1	196.9	3.1	3.6	2.0
	3	4.4	6.2	2.4	86.0	457.1	438.7	4.2	4.6	3.6
11:30	110	7.5	9.8	5.5	35.9	404.3	388.0	2.7	3.3	2.1
	30	6.1	7.7	4.9	54.4	420.7	404.3	2.9	3.3	2.5
	10	5.6	6.8	4.0	213.8	227.0	178.9	3.1	3.5	2.7
	3	4.3	5.8	3.2	87.1	462.4	435.5	4.2	4.6	3.7
11:35	110	7.9	9.4	6.1	37.0	409.1	384.8	2.7	3.3	2.2
	30	6.1	7.8	4.4	54.4	426.0	404.9	2.8	3.5	2.1
	10	5.7	7.8	3.7	183.7	203.2	176.8	3.1	3.5	2.3
	3	4.5	7.3	2.9	90.3	459.2	439.7	4.3	4.9	3.6

Table 3.1

This type data is available for seven of the experiments. The four points at each height represent an average.

The sound speed, sv, was calculated from

$$sv = 331.45 * \sqrt{(1 + T/273.15)} + w * \cos(\phi - 90) \quad (1)$$

where T is the temperature in Celcius, w is the wind speed, and ϕ is the angle that the wind is blowing relative to teh direction of propagation. A computer program performed a least square curve fit of the sound speed data to an equation of the form

$$sv = sv0 + m * \ln(z) \quad (2)$$

where sv0 is the sound speed at one meter, m is the slope of the curve, and z is the height above the ground. Typical values generated by this curve fit are given in Table 3.2.

Date	Run	Slope	c0
December 13, 1984	1.1	0.627	338.347
	1.2	0.663	338.609
	2.1	0.664	338.185
	4.1	0.948	338.770
January 11, 1985	2.1	-0.4876	325.320
	2.2	-0.4183	325.394

Table 3.2

A positive slope indicates that the atmosphere is downward refracting and a negative slope is upward refracting.

Also, in Appendix C there are weather data taken near the ground with a vertical array of thermocouples and temperature transducers. For the Sandusky site and June 23, 1985 Bondville site, both types of measurements were made. The measurements made with the thermocouple are indicated with the word trun and the other is run. Some measurements made at the Flatville site used only the thermocouple array.

IV. Relative Sound Pressure Level

The relative sound pressure level was calculated for each run. In all the measurements, a microphone placed close to the source served as a reference.

The relative sound pressure levels were computed from the distributions recorded by the MCA. The MCA output had the form of Figure 4.1. There are two distributions contained in the figure. The distribution marked B is the distribution of amplitude at an array microphone and D is the amplitude distribution at the reference microphone. The peaks marked A, C, and E are calibration points used to determine the dB levels of peaks B and D from a calibration tone put on each channel prior to every run. The difference in dB between the mode of the amplitude distribution recorded for the array and reference microphone is the relative sound pressure level. The difference is calculated as

$$R_{dB} = R_{odB} - 20 \log \left[\frac{xr_6 \cdot xc_i}{xc_6 \cdot xr_i}\right] - 20 \log \left[\frac{gc_6 \cdot gr_i}{gr_6 \cdot gc_i}\right] \tag{4.1}$$

where R_{odB} is the difference in dB between the reference calibration point and the array microphone's calculation point, xr_6 is the MCA channel number for the reference, xr_i is the MCA channel number of the mode for the ith array microphone, xc_6 is the MCA channel number for the calibration point for the ith array microphone, gc_6 is the gain of the amplifier for the reference channel, gc_i is the gain for the ith array microphone channel, gr_6 is the gain for reference, and gr_i is the gain for the ith array microphone.

The following example will show how the equation is used. Refering to Figure 4.2, peak A is xc_i, peak B is xr_i, peak D is xc_6, and peak E is xr_6. Reading from the graph

$$xc_i = 2.6$$
$$xr_i = 6.1$$
$$xc_6 = 3.2$$

and

$$xr_6 = 5.05.$$

From the data logs , $gc_6 = gr_6$ and $gc_i = gr_i$, also $R_{odB} = 44\text{-}84 = -40$ dB. Substituting these

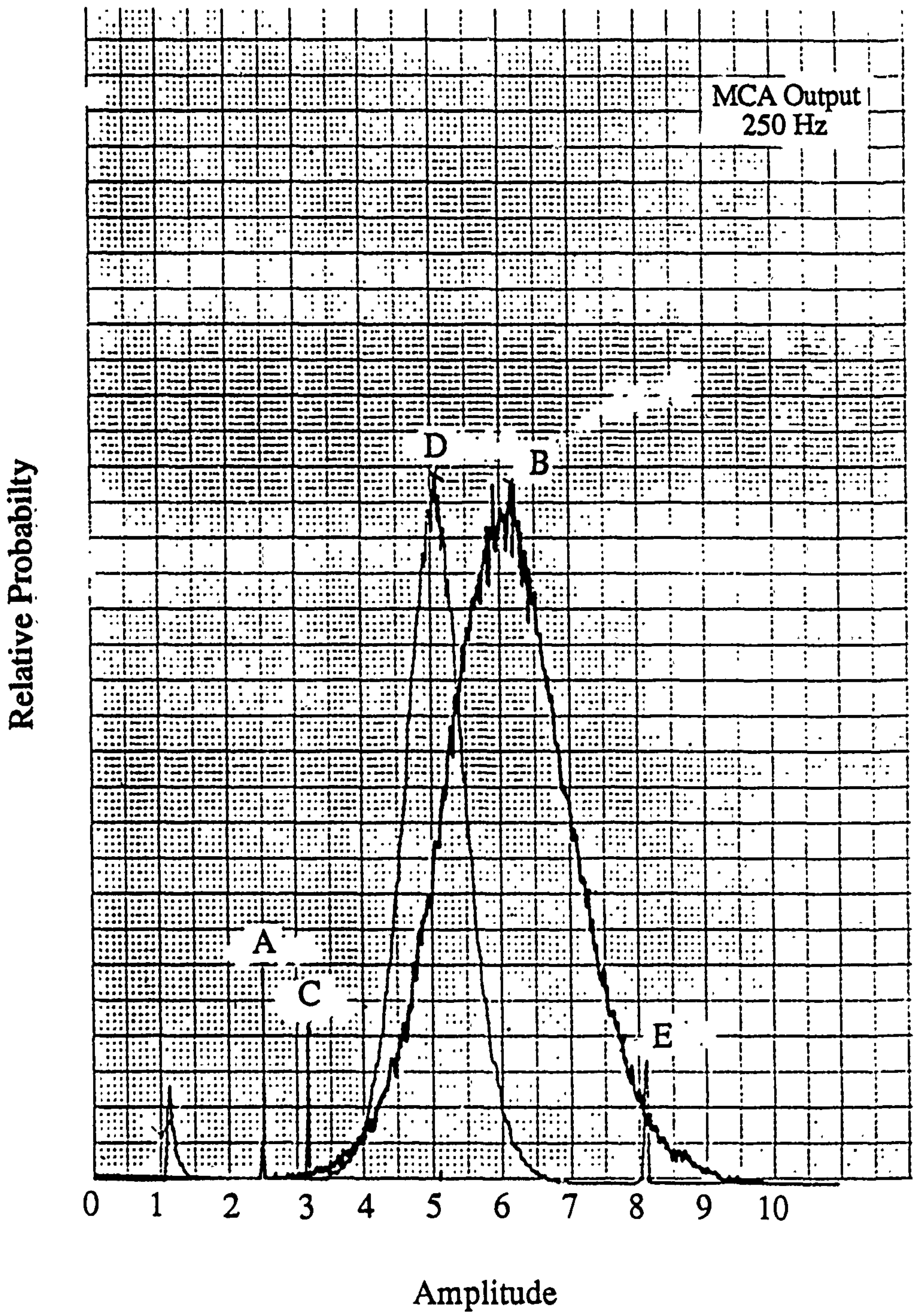

MCA Output
250 Hz
D
B
A
C
E
Relative Probabilty
0 1 2 3 4 5 6 7 8 9 10
Amplitude

Figure 4.1

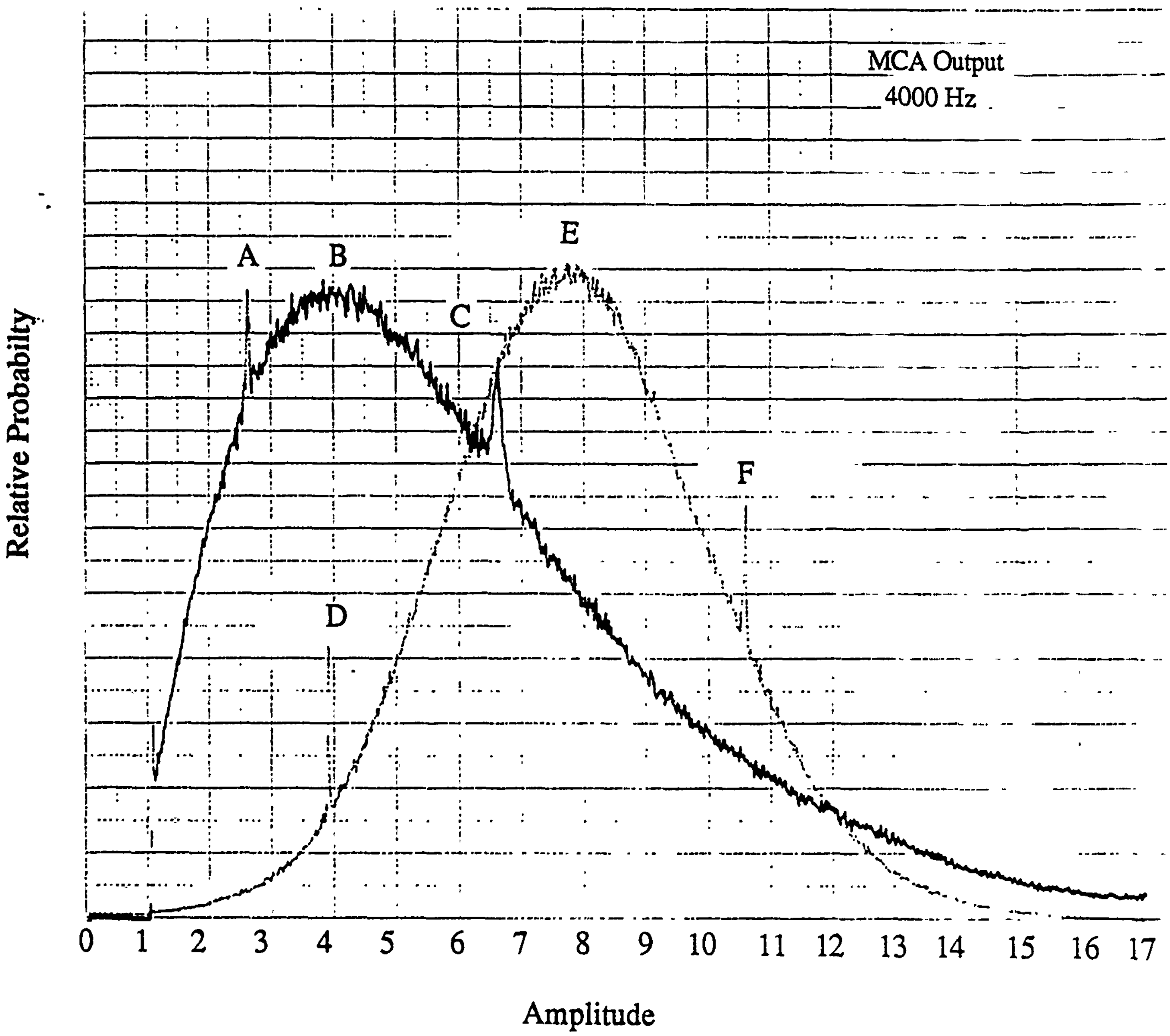

MCA Output
4000 Hz
A
B
C
D
E
F
Relative Probabilty
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
Amplitude

Figure 4.2

values into Eq. (4.1), R_{dB} is -42.28 dB. For most runs, the calibration gain is the same as for the run gain; however, there are a few runs where the gains are not the same. Plots of relative sound pressure level for every run are included as Appendix D.

V. Distribution Functions

Large eddies are formed in the atmosphere by instabilities in the thermal and viscous boundary layers near the ground. Further instability causes these eddies to break down into progressively smaller sizes until the energy is finally dissipated by viscosity in very small eddies. A statistical distribution of eddies of various sizes is therefore present in the atmosphere at all times

The fluctuations in the sound pressure level of a pure tone measured outdoors, however, are frequently much larger than would predict on the basis of sound propagation in an unbounded medium. The influence of turbulence on the sound field can be large when a boundary is present because the field above the boundary is critically dependent upon the phase relationships between the direct and reflected waves near a boundary, then fluctuations in phase and amplitude are not independent.

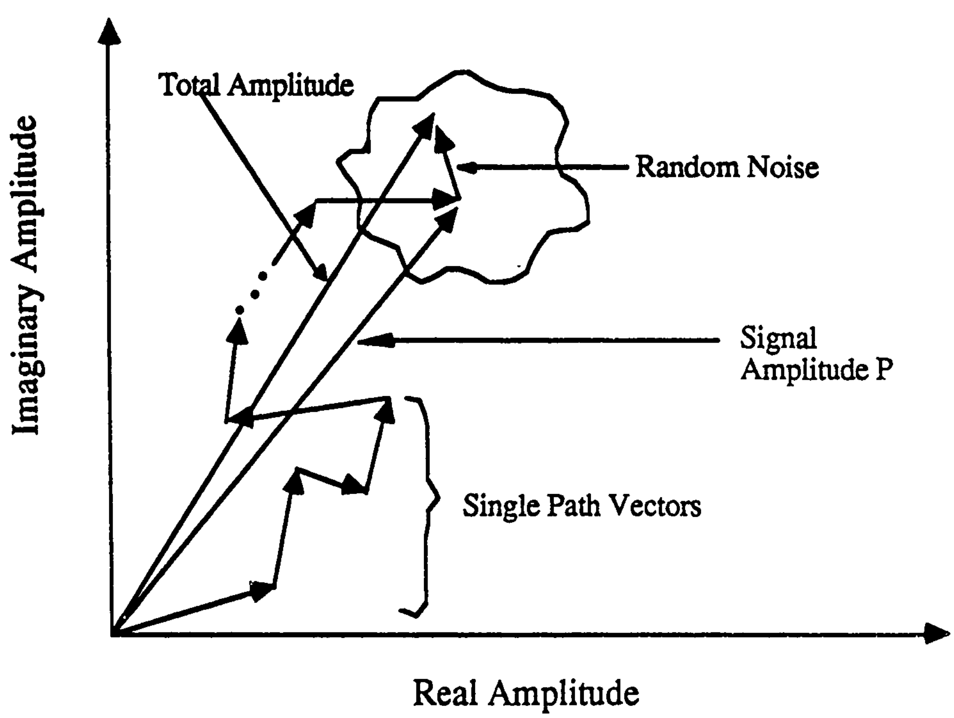


Figure 5.1

A polar representation of the signal is shown in Figure 5.1.

Note that the signal is composed of many single path vectors. These single path vectors are a mathematical representation of the various fluctuations in the phase and amplitude of the acoustical field as it propagates through the atmosphere. The total amplitude is a complex analysis representation of the complex signal amplitude P plus a complex random noise contribution. The random noise term takes into account the random nature of the fluctuations in the index of refraction of the turbules and size distribution of the turbules in the atmosphere over time. Any fluctuations in the total amplitude will depend upon the fluctuations in the single path phases and/or amplitudes, and the noise.[2]

In order to view the change in phase and amplitude over time, a series of scatter plots were made. A scatter plot is a representation of the changes in the complex amplitude over a fixed amount of time. Samples of these plots are provided in Figure 5.2 and Figure 5.3. The samples are from the January 11, 1985, run 2.1 data set.

These representative scatter plots appear as pictured in Figure 5.4.

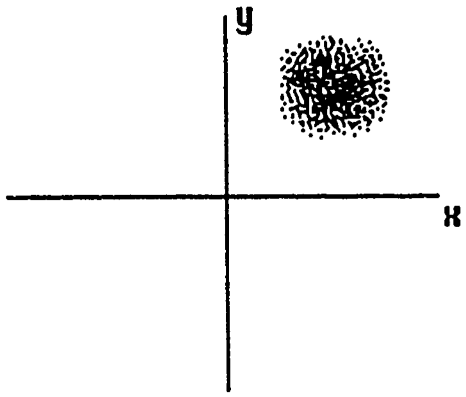


Figure 5.4

Assume that the real and imaginary parts have a Gaussian distribution about averages x and y. These assumptions give a Bivariant Normal Probability Function of the form

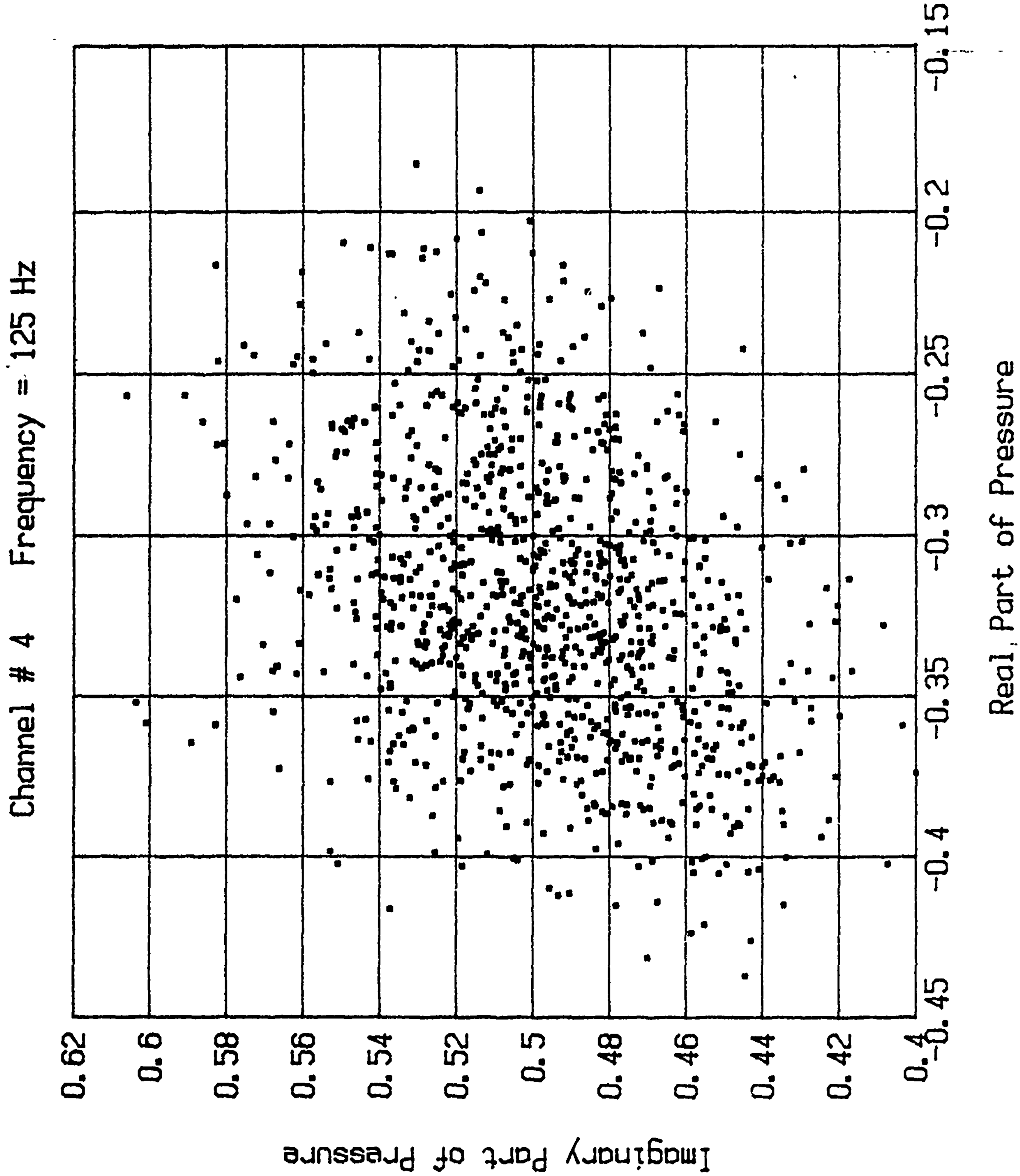
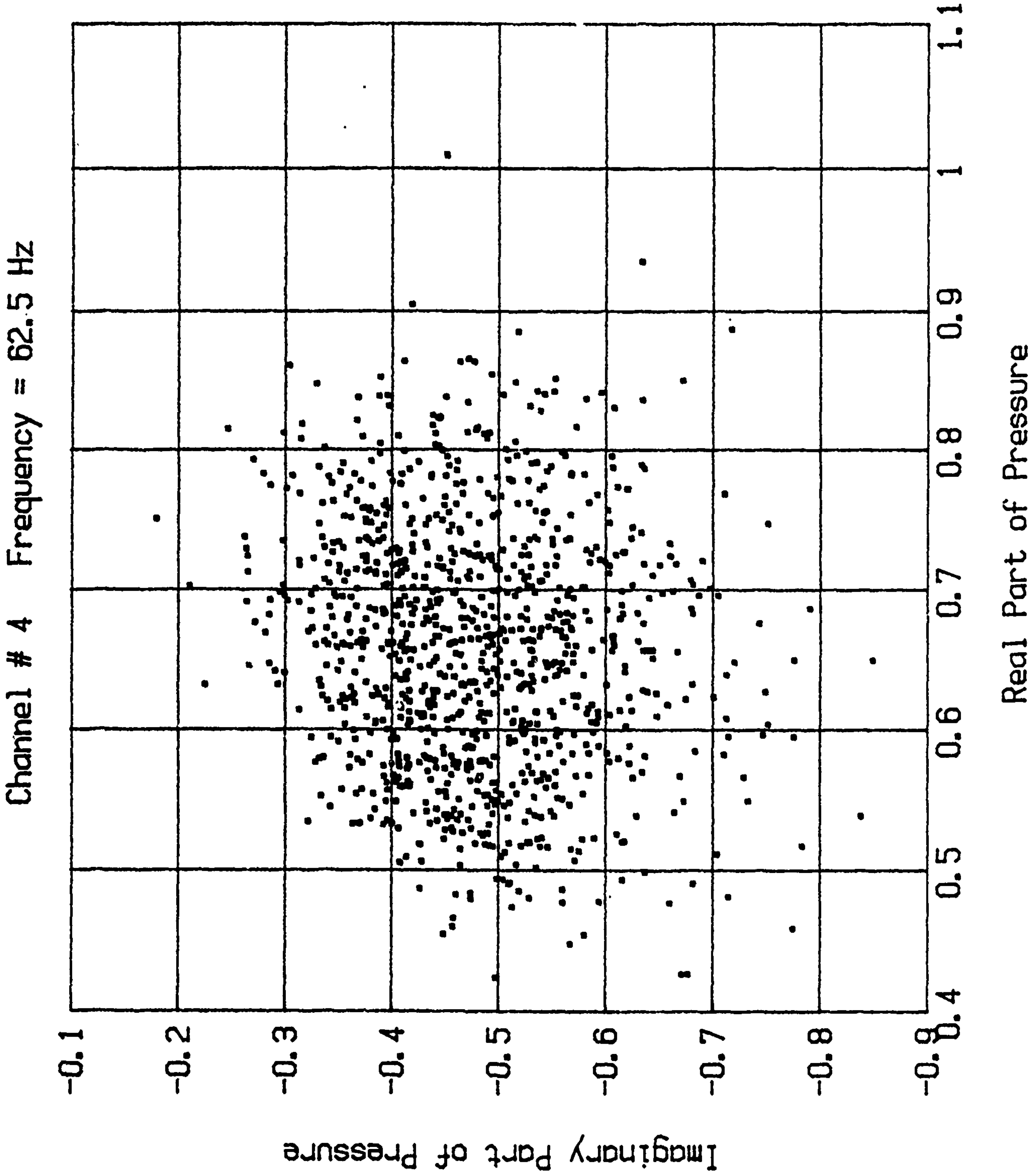

Channel # 4 Frequency = 125 Hz
Real Part of Pressure
Imaginary Part of Pressure
-0.15
-0.2
-0.25
-0.3
-0.35
-0.4
-0.45
0.62
0.6
0.58
0.56
0.54
0.52
0.5
0.48
0.46
0.44
0.42
0.4

Figure 5.2



Channel # 4 Frequency = 62.5 Hz
Real Part of Pressure
Imaginary Part of Pressure
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
-0.8
-0.9

Figure 5.3

$$P(x,y) = \frac{1}{2\pi\sigma^2} e^{-\left[(x-\bar{x})^2 + (y-\bar{y})^2\right]/2\sigma^2}$$

Converting to polar coordinates,

$$P(p) = p/\sigma^2 \, e^{-p^2+p_o^2/2\sigma^2} \, I_o \, pp_o/\sigma^2 \tag{5.1}$$

where p_o = the distance to the center of the distribution, $\left(\bar{x}_o^{\,2} + \bar{y}_o^{\,2}\right)^{1/2}$

σ = the standard deviation of the distribution

p = the distance to a point in the distribution, $(x^2 + y^2)^{1/2}$.

If $p_o/\sigma >> 1$, Eq. 5.1 reduces to

$$P(p) = \frac{1}{\sqrt{2\pi}\,\sigma} \sqrt{\frac{p}{p_o}} \, e^{-(p-p_o)^2/2\sigma^2}$$

which has the form of a skewed Gaussian. If $p_o/\sigma << 1$, Eq. 5.1 reduces to

$$P(p) = \frac{p}{\sigma^2} e^{-p^2/2\sigma^2}$$

which has the form of a Rayleigh distribution.

The mode of a distribution is the point where the maximum probability occurs. This is found by taking the derivative of the distribution function and equating it to zero. If this is carried out on Eq. (5.1), the result is

$$x^2 - x_o x \frac{I_1(x_o x)}{I_o(x_o x)} - 1 = 0$$

where $x = p/\sigma$ and x_o - p_o/σ.

Case 1: $x_o >> 1$ which is supposed to be Gaussian yields

$$\frac{I_1(x_o x)}{I_o(x_o x)} \rightarrow 1 \quad \text{as } x_o x \text{ becomes large}$$

$$x^2 - x_o x - 1 \approx 0$$

$x \approx x_o$ which means that

the mode of the distribution is p_o.

This result corresponds to that for a Gaussian distribution.

Case 2: $x_o << 1$ which is supposed to be Rayleigh yields

$$\frac{I_1(x_o x)}{I_o(x_o x)} \rightarrow 0 \quad \text{as } x_o x \text{ becomes small}$$

$$x^2 - 1 = 0$$

$x = 1$ which means that the mode of the distribution is σ. This corresponds to the behavior of a Rayleigh distribution.

Equation 5.1 was fit to the amplitude probability data from the MCA giving values for σ and P_o. Comparisons between measured and computed distributions are given at Appendix E. The program used to perform a least squares curfit to the data is in Appendix B.

VI. Structure Functions

For straight-line propagation (absence of refraction) a distance r through atmospheric turbulence, the log-amplitude and phase structure of functions in a plane perpendicular to the direction of propagation are defined as[1]

$$D_x(r,\rho) = \left\langle \left[x\left(\vec{r}+\vec{\rho}\right) - x\left(\vec{r}\right) \right]^2 \right\rangle \quad . \tag{6.1}$$

and

$$D_s(r,\rho) = \left\langle \left[\phi\left(\vec{r}+\vec{\rho}\right) - \phi\left(\vec{r}\right) \right]^2 \right\rangle \tag{6.2}$$

respectively, where ρ is the separation between receivers, x is the log amplitude, and ϕ is the phase. The mean square log-amplitude and phase fluctuations, $\langle x^2 \rangle = \langle(\ln A/A_o)^2\rangle$ and $\langle s^2 \rangle =$

$\langle(\phi - \phi_o)^2\rangle$, where A_o and ϕ_o are the amplitude and phase in the absence of turbulence, respectively.

If L is a measure of the scale of the turbulence, then in the case $(\lambda r)^{1/2} >> L$, the theory predicts that u is the rms fluctuations in the acoustic index of refraction

$$D_x(r,\rho) = 2\left[\langle x^2\rangle - B_x(\rho)\right] \tag{6.3}$$

$$D_s(r,\rho) = 2\left[\langle s^2\rangle - B_s(\rho)\right] \tag{6.4}$$

where

$$\langle x^2\rangle = \langle s^2\rangle = \left(\frac{\sqrt{\pi}}{2}\right)\langle u^2\rangle k^2 rL \tag{6.5}$$

u is the rms fluctuation in the acoustic index of refraction, k is the propagation constant, and $B_x(\rho)$ and $B_s(\rho)$ are, respectively, the covariances of the log-amplitude and phase fluctuations.[5]

$$\frac{B_x(\rho)}{\langle x^2\rangle} = \frac{B_s(\rho)}{\langle s^2\rangle} = \frac{\Phi(\rho/L)}{\rho/L} \tag{6.6}$$

where

$$\Phi(\rho/L) = \int_0^{\rho/L} e^{-u^2} du.$$

In practice Eq. (6.5) agrees with experimental results for $\langle s^2\rangle$. However, measurements show that $\langle x^2\rangle << \langle s^2\rangle$ and, in addition, the log-amplitude fluctuations quickly saturate. Substituting Eq. (6.6) and (6.5) for x into Eq. 6.3 yields

$$D_x(r,\rho) = \sqrt{\pi}\langle u^2\rangle k^2 rL\left[1 - \frac{\Phi(\rho/L)}{\rho/L}\right]. \tag{6.7}$$

The extraction of the mean square amplitude and phase fluctuations requires processing of the recorded acoustical signals. The analysis requires the assumption that the phase and amplitude sequences represent samples of random processes where ensemble averages equal time averages (Taylor's Frozen Turbulence Hypothesis). Let i, j refer to two microphones in a plane

perpendicular to the direction of propagation. Then the log-amplitude for the ith microphone at the nth time sample is

$$x_{in} = \ln\left(\frac{A_{in}}{A_{io}}\right) \tag{6.8}$$

where A_{in} denotes the amplitude and

$$A_{io} = \frac{1}{N}\sum_{n=1}^{N} A_{in} \tag{6.9}$$

is the average amplitude over N samples. The amplitude structure function, Eq. (6.1), is then computed from

$$D_x = \frac{1}{N}\sum_{n-1}^{N}(x_{in} - x_{jn})^2. \tag{6.10}$$

Similarly the phase structure function, Eq. (6.2) is obtained from

$$D_s = \frac{1}{N}\sum_{i=1}^{N}(\phi_{in} - \phi_{jn})^2 - \left[\frac{1}{N}\sum_{n=1}^{N}(\phi_{in} - \phi_{jn})\right]^2 . \tag{6.11}$$

Equation (6.11) is written in its more general form where the last term accounts for the fact that in experimental practice the mean phase difference of the measured signals may not be zero.

There are three programs which are used for this task. They are called phaz, phase, and turbl. The purpose of the program phaz was discussed in section 2. The program phase takes the output from phaz and calculates the phase structure function using Eq. (6.11). The program turbl is used to perform the calculations needed to arrive at the log-amplitude structure function.

The output from phase and turbl was plotted against transverse separation of the microphones. These plots can be found in Appendix F. The solid line is the theoretical curve based on Eq. (6.7) using the values from Table 6.1 for $\langle u^2 \rangle$ and L measured according to the method outlined by Johnson, Raspet, and Bobak.[4]

	1/11/85		12/13/84			
	Run 2.1	Run 2.2	Run 1.1	Run 1.2	Run 2.1	Run 4.1
$\langle u^2 \rangle$	1.6×10^{-6}	2.56×10^{-6}	8.8×10^{-6}	11.7×10^{-6}	8.41×10^{-6}	15.4×10^{-6}
L	15 m	0.82 m	15.0 m	15.0 m	15.0 m	2.17 m

Table 6.1

References

1. G.A. Daigle, J. E. Piercy, and T.F.W. Embleton, "Effects of Atmospheric Turbulence on the Interface of Sound Waves Near a Hard Boundary," J. Acoust. Soc. Am. 64, 622-630 (1978).

2. Peter N. Mikhalevsky, "Characteristics of CW Signals Propagated Under the Ice in the Arctic," J. Acoust. Soc. Am. 70, 1717-1722 (1981).

3. S. F. Clifford and R.J. Lataitis, "Turbulence Effects on Acoustic Propagation Over a Smooth Surface," J. Acoust. Soc. Am. 73, 1545-1550 (1983).

4. M.A. Johnson, R. Raspet, and M.T. Bobak, "A Turbulence Model for Sound Propagation from an Elevated Source Above Level Ground," J. Acoust. Soc. Am. 81, 638-646 (1987).

5. V.N. Karavinikov, "Fluctuations of Amplitude and Phase in a Spherical Wave," Akast. Zh. 3, 175-186 (1957).

APPENDICES

A. Geometries
This appendix contains the geometrical configuration for all of the experimental runs performed.

B. Programs
This appendix contains the computer programs used to analyze or compute the results obtained.

C. Weather
Contains the plots of the different meteorological parameters measured during the experiment.

D. Relative Sound Pressure Levels
Contains the plots of the relative sound pressure level for each microphone and each experiment.

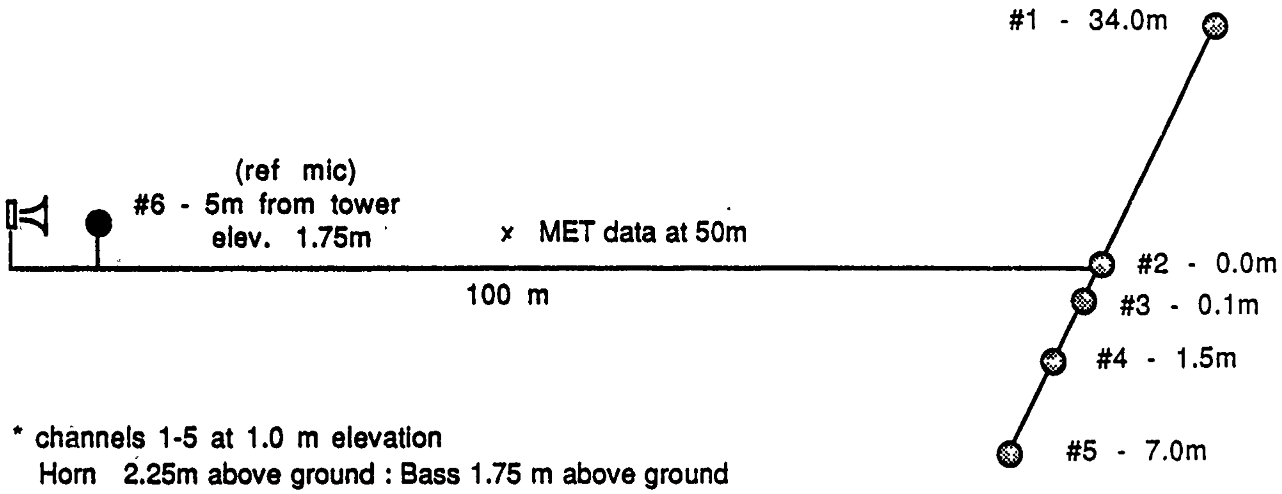
E. Distribution
Contains the MCA data that has been compared with the bivariant normal probability function.

F. Structure Function
Has the comparisons between the structure functions calculated from the data and Daigle's theoretical structure function.

APPENDIX A

Geometrical configuration for all of the experimental runs performed.

GEOMETRY
JUNE 19-20, 1984



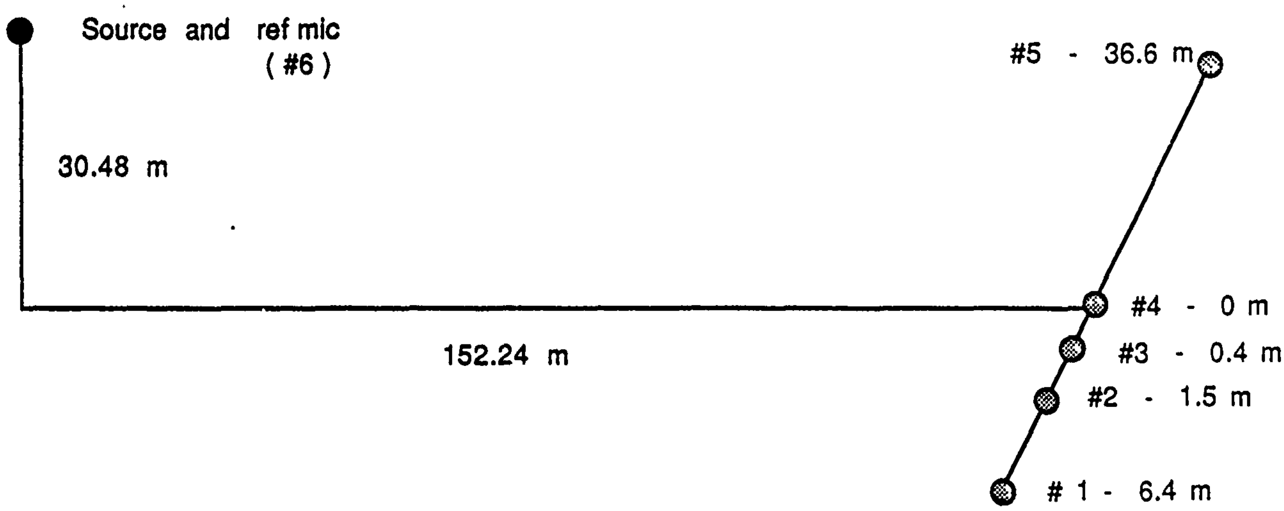
* channels 1-5 at 1.0 m elevation
Horn 2.25m above ground : Bass 1.75 m above ground

TRANSVERSE DISTANCES

1/2	34.0m
1/3	34.1
1/4	35.5
1/5	41.0
2/3	0.1
2/4	1.5
2/5	7.0
3/4	1.4
3/5	6.9
4/5	5.5

Bondville, Ill.
Dec. 13, 1984
Runs 1.1 & 1.2

GEOMETRY

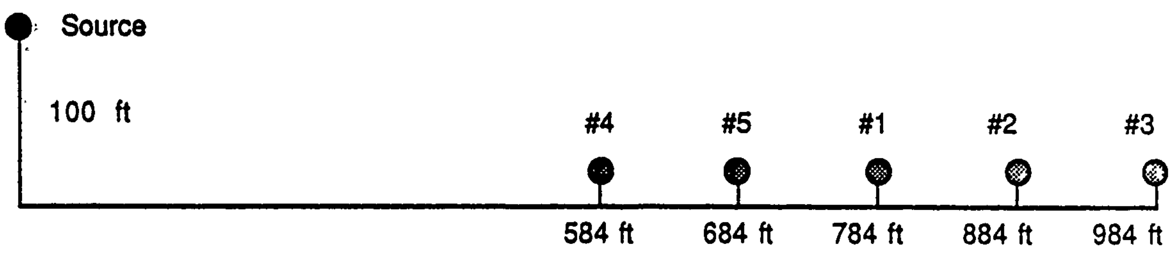


Transverse Distances

1/2	5.0 m
1/3	6.1 m
1/4	6.4 m
1/5	43.0 m
2/3	1.1 m
2/4	1.4 m
2/5	38.0 m
3/4	0.3 m
3/5	36.9 m
4/5	36.6 m

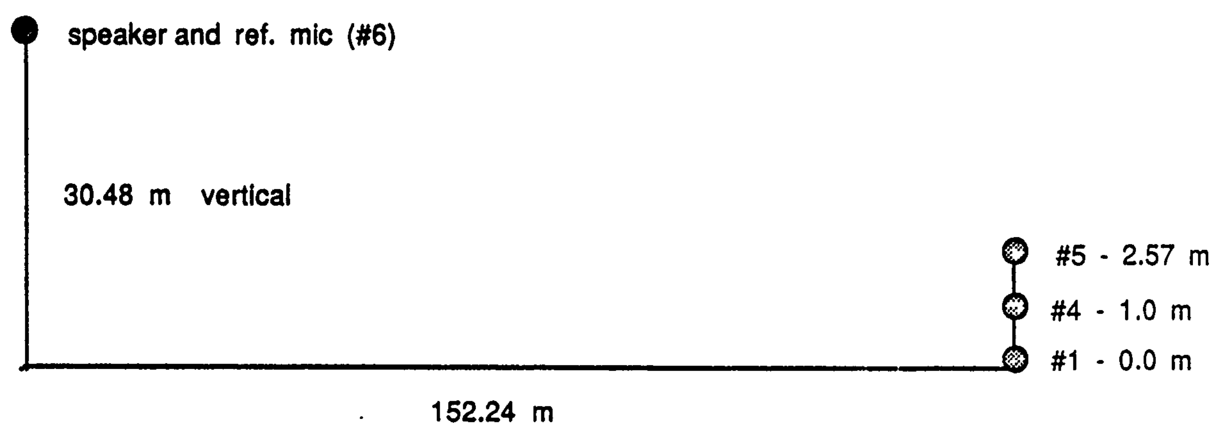
Bondville, Ill.
Dec, 13 1984
Run 2.1

Geometry



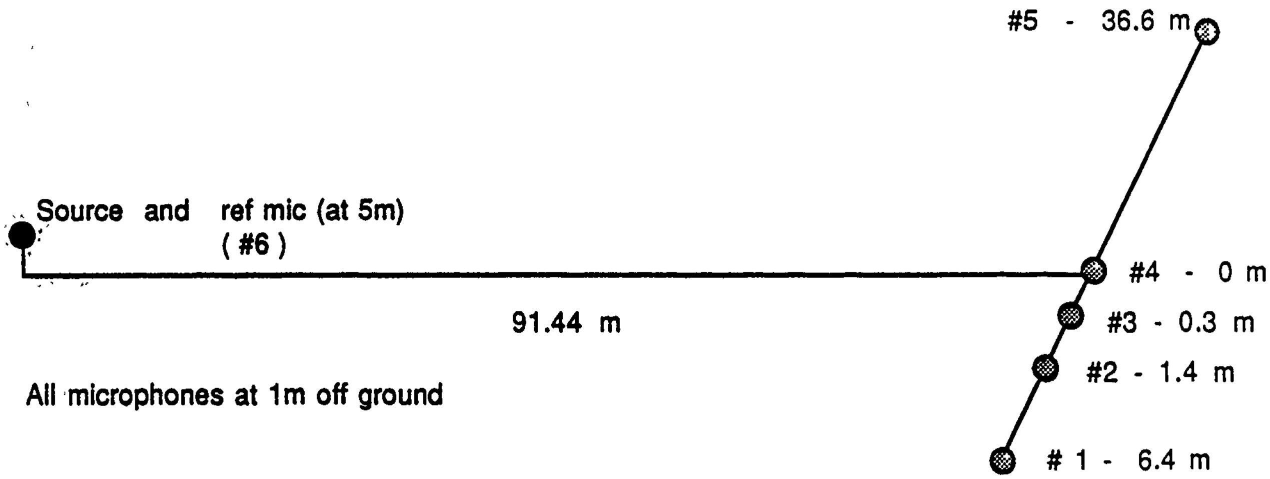
BONDVILLE, ILL.
DEC. 13, 1984
RUN 3.1

GEOMETRY



Bondville Ill.
Dec 13, 1984
Run 4.1

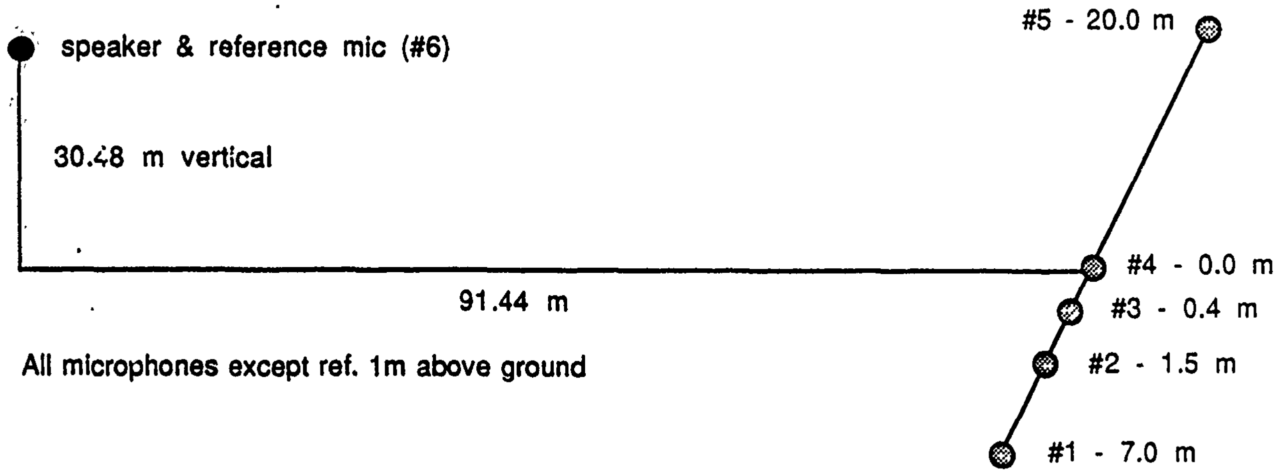
GEOMETRY



Transverse	Distances
1/2	5.0 m
1/3	6.1 m
1/4	6.4 m
1/5	43.0 m
2/3	1.1 m
2/4	1.4 m
2/5	38.0 m
3/4	0.3 m
3/5	36.9 m
4/5	36.6 m

BONDVILLE, ILL.
JAN. 11, 1985
RUN 2.1

GEOMETRY



All microphones except ref. 1m above ground

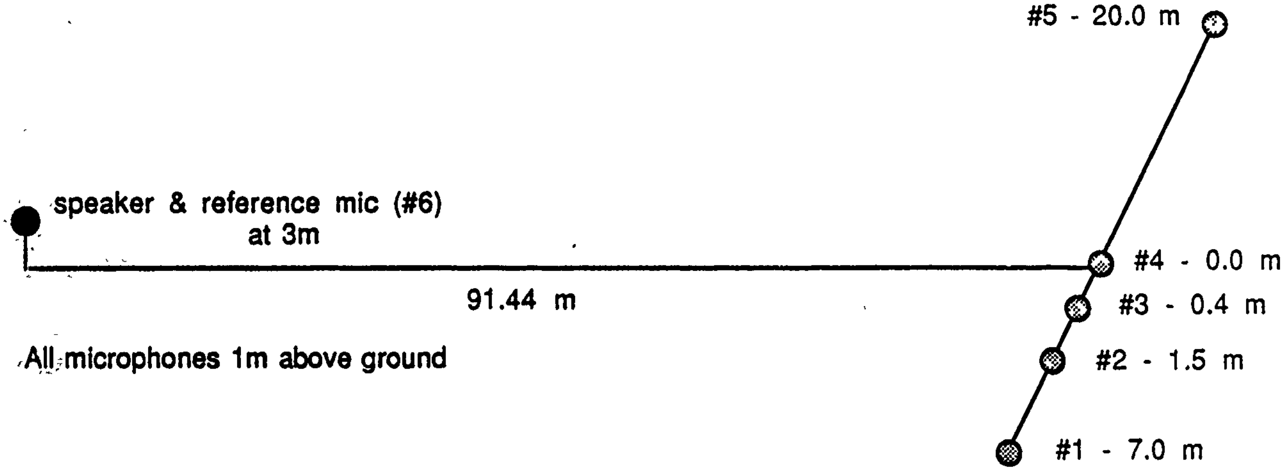
Transverse Distances

1/2	-	5.5 m
1/3	-	6.6
1/4	-	7.0
1/5	-	27.0
2/3	-	1.1
2/4	-	1.5
2/5	-	21.5
3/4	-	0.4
3/5	-	20.4
4/5	-	20.0

Ch. #	Mic. #
1	2
2	3
3	4
4	7
5	5
6	ref
7	voice

BONDVILLE, ILL.
JAN. 11, 1985
RUN 2.2

GEOMETRY



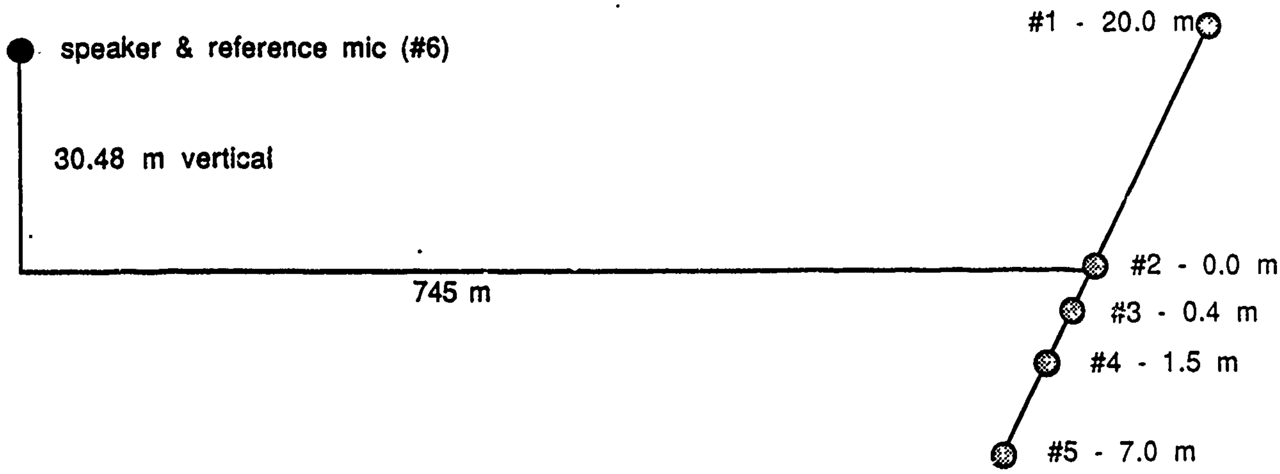
Transverse Distances

1/2	-	5.5 m
1/3	-	6.6
1/4	-	7.0
1/5	-	27.0
2/3	-	1.1
2/4	-	1.5
2/5	-	21.5
3/4	-	0.4
3/5	-	20.4
4/5	-	20.0

Ch. #	Mic. #
1	2
2	3
3	4
4	7
5	5
6	ref
7	voice

BONDVILLE, ILL.
JULY 23, 1985
RUN #1

GEOMETRY

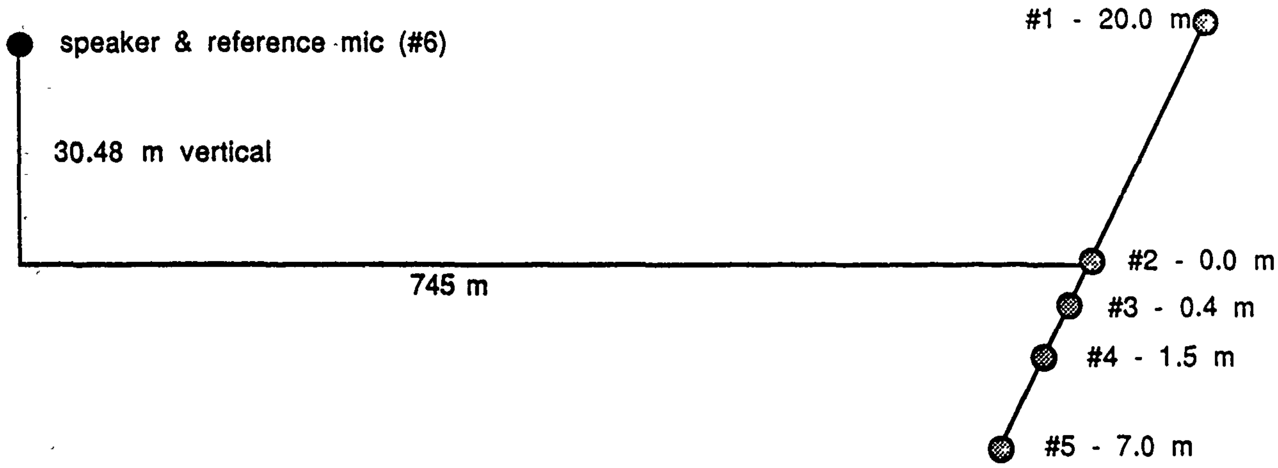


Transverse	Distances
1/2	20.0 m
1/3	20.4
1/4	21.5
1/5	27.0
2/3	0.4
2/4	1.5
2/5	7.0
3/4	1.1
3/5	6.6
4/5	5.5

Ch. #	Mic. #
1	7
2	5
3	4
4	3
5	2
6	?
7	voice

BONDVILLE, ILL.
JULY 23, 1985
RUN # 2

GEOMETRY

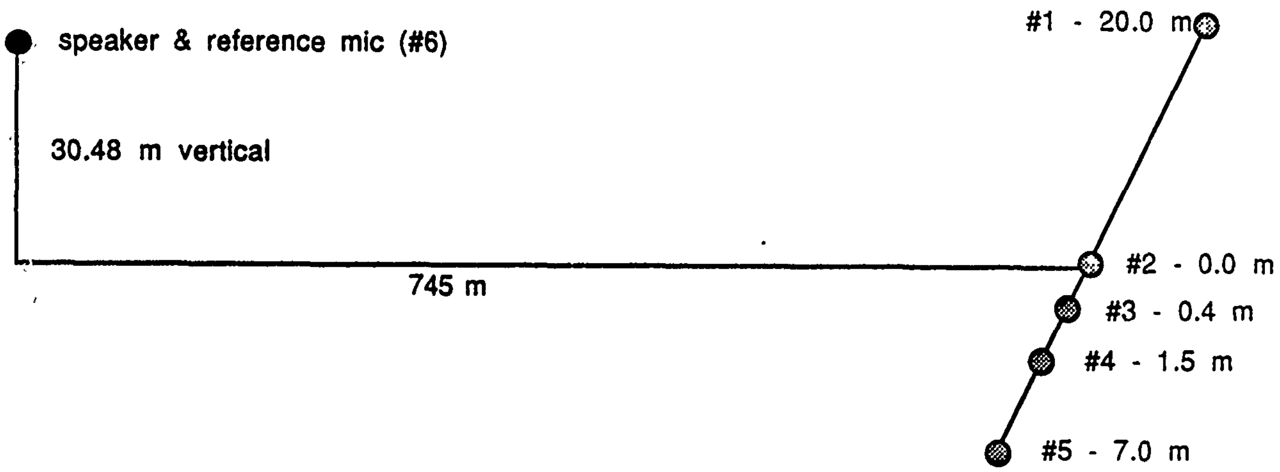


Transverse	Distances
1/2	20.0 m
1/3	20.4
1/4	21.5
1/5	27.0
2/3	0.4
2/4	1.5
2/5	7.0
3/4	1.1
3/5	6.6
4/5	5.5

Ch. #	Mic. #
1	7
2	5
3	4
4	3
5	2
6	?
7	voice

BONDVILLE, ILL.
JULY 23, 1985
RUN # 3

GEOMETRY

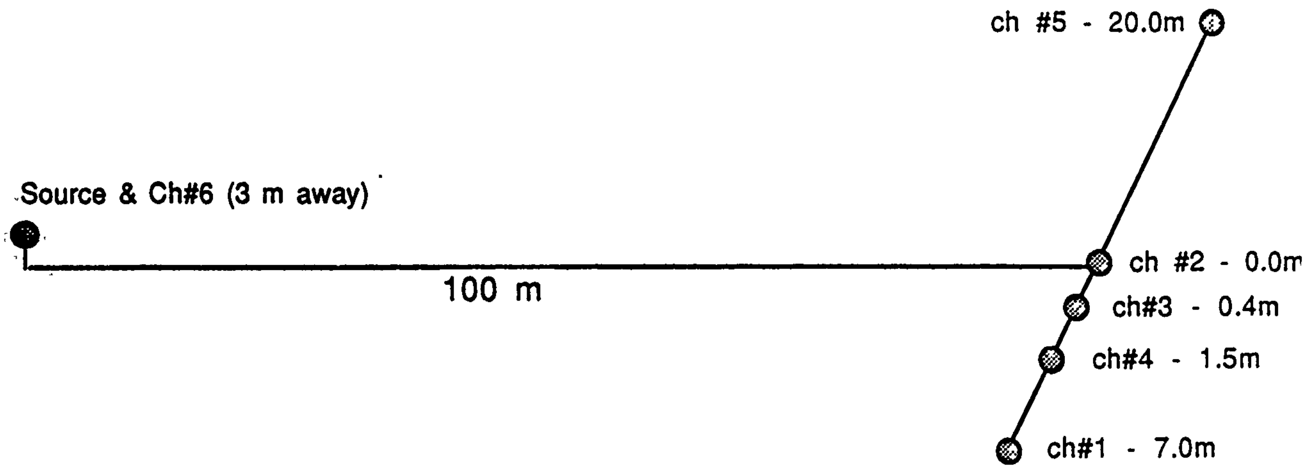


Transverse	Distances
1/2	20.0 m
1/3	20.4
1/4	21.5
1/5	27.0
2/3	0.4
2/4	1.5
2/5	7.0
3/4	1.1
3/5	6.6
4/5	5.5

Ch. #	Mic. #
1	7
2	5
3	4
4	3
5	2
6	?
7	voice

BONDVILLE, ILL.
JULY 25, 1985
RUN #1
RUN #2

GEOMETRY



Horn 2 m above ground: Bass 1.36 m above ground
Source 2 m above ground
All microphones 1 m above ground

TRANSVERSE DISTANCES

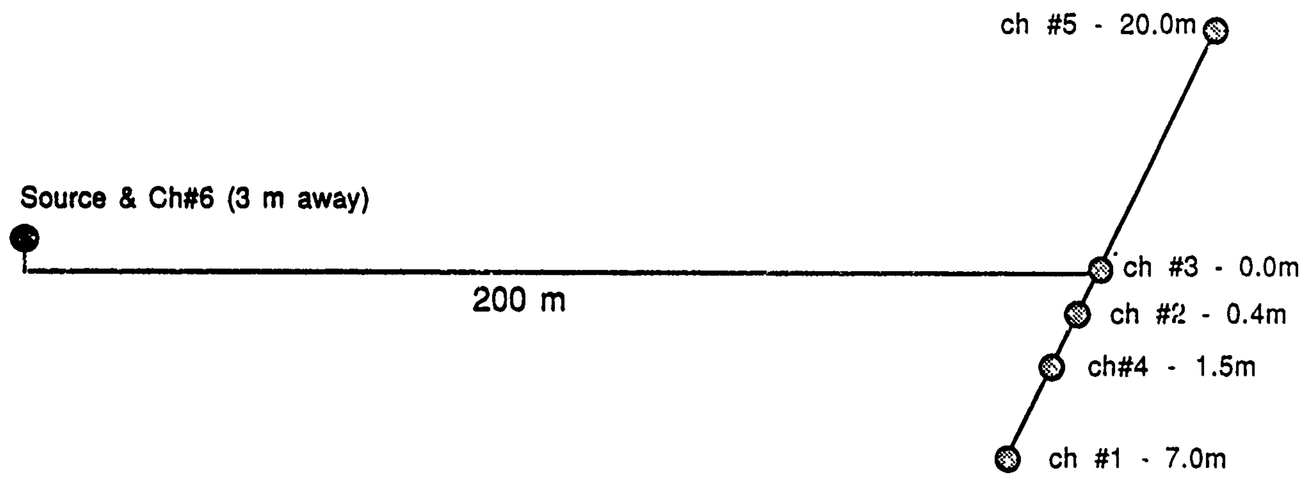
ch#/ch#	distance
1/2	7.0m
1/3	6.6
1/4	5.5
1/5	27.0
2/3	0.4
2/4	1.5
2/5	20.0
3/4	1.1
3/5	20.4
4/5	21.5

AKG MICROPHONES

mic#	ch#
7	1
blank	2
4	3
AKG standard	4
2	5
Richard's	6

BONDVILLE, ILL.
JULY 25, 1985
RUN #3

GEOMETRY



Horn 2 m above ground: Bass 1.36 m above ground
Source 2 m above ground
All microphones 1 m above ground
*** notice ch#2 and ch#3 switched ***

TRANSVERSE DISTANCES

ch#/ch#	distance
1/2	6.6m
1/3	7.0
1/4	5.5
1/5	27.0
2/3	0.4
2/4	1.1
2/5	20.4
3/4	1.5
3/5	20.0
4/5	21.5

AKG MICROPHONES

mic#	ch#
7	1
blank	2
4	3
AKG	4
2	5
Richard's	6

APPENDIX B

Computer programs used to analyze or compute the results obtained.

```
c-----------------------------------------------------------c
c  this program will read the weather data and convert to svp  c
c  then perform the statistics                                 c
c-----------------------------------------------------------c
c
        character*32 nfile
        real m(3),b(3),mavg
        dimension w(4,10),d(4,10),t(4,10),sv(4,10),ad(4),x(4)
        dimension xs(4),x2s(4),ys(4),y2s(4),xys(4),n(4),h(4),r2(3)
        print*,"Input data file"
        read*,nfile
        open(1,file=nfile,status="old")
        rewind 1
        h(1)=110
        h(2)=30
        h(3)=10
        h(4)=3
        x(1)=3.5571
        x(2)=2.3086
        x(3)=1.1147
        x(4)=-0.09431
        do 100 i=1,3
           nd=0
           ad(i)=0.0
           do 200 j=1,4
              read(1,*) w(i,j),d(i,j),t(i,j)
              ad(i)=d(i,j)+ad(i)
              if (d(i,j).ne.0.0) then
                 nd=nd+1
              endif
200        continue
           ad(i)=ad(i)/float(nd)
           do 400 j=1,4
              sv(i,j) = 331.5 * sqrt(1 + t(i,j) / 273.15)
     +                 + w(i,j) * cos((ad(i) - 90) / 57.295779)
              if (t(i,j).eq.0.0.and.w(i,j).eq.0.0) then
              sv(i,j)=0.0
              endif
              write(6,500) h(j),sv(i,j)
400           continue
100     continue
500     format(5x,2f8.2)
c
c-----------------------------------------------------------c
c  this section starts the statistical portion of the         c
c  program.  first we calculate the sums for each subset.     c
c-----------------------------------------------------------c
c
        do 700 i=1,3
           xs(i)=0.0
           x2s(i)=0.0
           ys(i)=0.0
           y2s(i)=0.0
           xys(i)=0.0
           n(i)=4
```

```
          do 800 j=1,4
             if (sv(i,j).eq.0.0) go to 750
             xs(i)=xs(i)+x(j)
             x2s(i)=x2s(i)+x(j)*x(j)
             ys(i)=ys(i)+sv(i,j)
             y2s(i)=y2s(i)+sv(i,j)*sv(i,j)
             xys(i)=xys(i)+sv(i,j)*x(j)
750             if (sv(i,j).eq.0.0) then
                n(i)=n(i) - 1
             endif
800          continue
          m(i)=(float(n(i))*xys(i)-xs(i)*ys(i))/
     a                (float(n(i))*x2s(i)-xs(i)*xs(i))
          b(i)=(ys(i)-m(i)*xs(i))/float(n(i))
          r2(i)=(float(n(i))*b(i)*ys(i)+float(n(i))*m(i)*xys(i)
     a          -ys(i)*ys(i))/(float(n(i))*y2s(i)-ys(i)*ys(i))
700    continue
       if (n(1).eq.n(2).and.n(1).eq.n(3)) go to 1100
       write(6,"('unequal data sets, check input')")
       go to 2000
1100   continue
c-----------------------------------------------------c
c  this section calculates the combined statistics   c
c-----------------------------------------------------c
       mavg=0.0
       bavg=0.0
       r2avg=0.0
       nm=0
       do 1300 i=1,3
          mavg=mavg+m(i)
          bavg=bavg+b(i)
          r2avg=r2avg+r2(i)
          nm=nm+1
1300   continue
       mavg=mavg/float(nm)
       bavg=bavg/float(nm)
       r2avg=r2avg/float(nm)
       write(6,1500) mavg,bavg,r2avg
1500   format(1x,"mavg=",f8.4,"  bavg=",f8.4,"  r2avg=",f8.5)
2000   stop
       end
```

```
c       ***************************************************
c       ***************************************************
c       **********              phaz.f           **********
c       **********   Finds the amplitude and phase    **********
c       **********      difference between two     **********
c       **********       different channels.       **********
c       ***************************************************
c       ***************************************************
c
c
        program phaz

        integer hiwin, i, iblk, ch1, ch2, dgfreq, ipts, j
        integer k, lowin, n1, n2, xcycl1, xcycl2, nk
        integer npass, numch
        integer*2   data(1024,2)
        real     area1, area2, bias1, bias2, freq
        real     data1(1024), data2(1024), hper, lastp1, lastp2
        real     noise1, noise2, period, phasinc, phastot, pi, sumtim
        real     tol, tsampl, timkp1, timkp2
        real     time(1024,2), phase(2048), ampl(1024,2)
        parameter (pi=3.141592654, tol=0.15)

        open(20,file='/data/adc.dat',access='direct',status='old',recl=2

        open(1,file='/data/holder',status='old')
        open(2,file='/data/ampl.dat')
        open(4,file='/usr/data/phase.dat')
        open(3,file='/usr/data/freq.dat')
        rewind 1
        read(1,*),numch
        read(1,*),dgfreq
        read(1,*),ipts
        read(1,*),ch1
        read(1,*),ch2

         tsampl = 1 / float(dgfreq)
         freq = float(dgfreq / 12)
         hper = 1 / ( 2 * freq )
         npass = ipts / (numch * 1024)
         xcycl1 = 0
         xcycl2 = 0
         lastp1 = 0.
         lastp2 = 0.
         area1 = 0.
         area2 = 0.
         phastot = 0.
         iblk = 0
         nk = 0
c        ****************************************
c        ** Read In 1024 Points From Each Channel **
```

```
c      ** Skip First Hundred Points              **
c      ******************************************
10     lowin = ch1 + nk * numch + 100 * numch
       hiwin = ch1 + (nk + 1023) * numch + 100 * numch
       j = 1
       do 2 i=lowin,hiwin,numch
          read(20,rec=i),data(j,1)
          read(20,rec=i+ch2-ch1),data(j,2)
          j = j + 1
 2     continue
       nk = nk + 1024
c      ***********************************************
c      *****  Determine DC Bias  (Just Average Over   *****
c      *****  Complete Cycles, 1020 instead of 1024)  *****
c      ***********************************************
       bias1 = 0.0
       bias2 = 0.0
       do 15 i=1,1020
          bias1 = bias1 + float(data(i,1))
          bias2 = bias2 + float(data(i,2))
15     continue
       bias1 = bias1 / 1020.
       bias2 = bias2 / 1020.
c      *******************
c      ** Remove DC Bias **
c      *******************
       do 20 i = 1, 1024
          data1(i) = data(i,1) - bias1
          data2(i) = data(i,2) - bias2
20     continue
c
       n1=xcycl1+1
       n2=xcycl2+1
c      ***************************************************
c      *****  Determine Local Zeroes For Channels 1 & 2  *****
c      ***************************************************
       call cycle(1,data1,n1,lastp1,timkp1,area1,time,ampl,
      +                tsampl)
       n1=n1-1
       call cycle(2,data2,n2,lastp2,timkp2,area2,time,ampl,
      +                tsampl)
       n2=n2-1
c      ****************************************
c      *****   Zero Crossings are now in   *****
c      *****   Time Array.  Compute Period *****
c      ****************************************
       sumtim=0.0
       do 65 i=1,n1
          sumtim=sumtim+time(i,1)
65     continue
```

```
      period=2.*sumtim/n1
      freq=1.0/period
c     ********************************************************
c     *****   Check Signal to Noise Ratio for this Block   *****
c     ********************************************************
      noise1=float(nzer1)/float(n1)*100.
      noise2=float(nzer2)/float(n2)*100.
      if (noise1.lt.5.) then
          if (noise2.lt.5.) then
c            ****************************
c            *****   Compute Phase   *****
c            ****************************
                phasinc=(time(1,1)-time(1,2))/period*360.
                phase(1)=phastot+phasinc
                k=min0(n1,n2)
                do 70 i=2,k
                   phasinc=(time(i,1)-time(i,2))/period*360.
                   phase(i)=phase(i-1)+phasinc
70              continue
                do 75 i=1,k
                   if (iblk.eq.0.and.i.eq.1) go to 75
                   write(2,*)ampl(i,1),ampl(i,2)
                   write(4,*)phase(i)
75              continue
                phastot=phase(k)
             else
                print*,'Ch 2. Too Noisy. Skip Data. Hold Phase.'
          endif
        else
          print*,'Ch1. Too Noisy. Skip Data. Hold Phase.'
      endif
      iblk=iblk+1
      if (iblk.ge.npass) go to 95
      xcycl1 = 0
      xcycl2 = 0
      if (n2.gt.n1) then
c        *********************************
c        *****   More Cycles on Ch 2  *****
c        *********************************
         xcycl2=n2-n1
         do 85 i=1,xcycl2
            time(i,2)=time(n1+i,2)
            ampl(i,2)=ampl(n1+i,2)
85       continue
      endif
      if (n1.gt.n2) then
c        *********************************
c        *****   More Cycles on Ch 1  *****
c        *********************************
         xcycl1=n1-n2
```

```
        do 90 i=1,xcycl1
           time(i,1)=time(n2+i,1)
           ampl(i,1)=ampl(n2+i,1)
90      continue
      endif
      go to 10
95    write(3,*) freq
      stop
      end
c
c
c     ************************************
c     ** Finds the Zero Crossing Between **
c     **      (0,y1) And (dx,y2)         **
c     ************************************
      real function zerox(y1,y2,dx)
      real    y1, y2, slope, dx
      slope = (y2 - y1) / dx
      zerox = - y1 / slope
      return
      end
c
c
      subroutine cycle(nch,data,ncyc,lastpt,timeKp,area,
     +                  time, ampl, tsampl)
c     ***************************************************
c     **  Processes Cycles in Each Block of 1024 Points  **
c     **  Records Time and Amplitude of each 1/2 cycle   **
c     ***************************************************
c
      integer   i, ncyc, nch
      real      ampl(1024,2), time(1024,2), data(1024)
      real      aftime, zerox, thispt, lastpt, prod, timeKp, tsampl
      real      pi, area
      parameter (pi=3.141592654)
c
      do 10 i=1,1024
         thispt=float(data(i))
         prod = thispt*lastpt
         if (prod.ge.0) then
c           ******************************
c           *****  Same Side of Zero  *****
c           ******************************
            timeKp = timeKp + tsampl
            area = area + tsampl*(thispt+lastpt)/2
            if (thispt.eq.0) then
c              ******************************
c              *****  Hit Zero Exactly  *****
c              ******************************
               time(ncyc,nch) = timeKp
```

```
                  ampl(ncyc,nch) = (pi*area)/(2*timeKp)
                  ncyc = ncyc + 1
                  timeKp = 0
                  area = 0
               endif
            else
c              ********************************************
c              *****  Crossed Zero -- End of 1/2 Cycle  *****
c              ********************************************
               aftime = zerox(lastpt,thispt,tsampl)
               timeKp = timeKp + aftime
               area = area + lastpt*aftime/2
               time(ncyc,nch) = timeKp
               ampl(ncyc,nch) = (pi*area)/(2*timeKp)
               ncyc = ncyc + 1
c              *******************************************
c              *****  Get Prepared For Next 1/2 Cycle  *****
c              *******************************************
               timekp = tsampl - aftime
               area = thispt*timekp/2
            endif
c           **************************************
c           *****  Remember This Data Point  *****
c           **************************************
            lastpt = thispt
10       continue
c
         return
         end
```

```
c   ******************************************************
c   ******************************************************
c   **********              phase.f            **********
c   **********    Finds the average phase and   **********
c   **********    the phase structure function  **********
c   **********      for a pair of channels.     **********
c   ******************************************************
c   ******************************************************
c
c
      program phase
      dimension tot(1000),tot2(1000),ds(1000)
      integer*4 num(1000)
    2 format(15x,e13.6,12x,e13.6)
    3 format(1x,/,25x,'Corrected Results ')
    4 format(10x,'Overall Avg Phase =',e13.6,3x,'Avg Structure =',e13.6)
    7 format(25x,'Raw Data')
    1 format(23x,'Very Raw Data')
    8 format(10x,'Phase Structure Function',6x,'Average Phase',/,
     8     18x,'(rad**2)',18x,' (rad) ')
    9 format(10x,'Total Phase Structure = ',e13.6)
   16 format(10x,'Frequency = ',f11.6)
      open(4,file='/usr/data/phase.dat',status='old')
      open(1,file='/usr/data/freq.dat',status='old')
      open(2,file='/usr/data/ph')
      open(3,file='/usr/data/cph')
      open(5,file='/usr/data/raw')
      rewind 1
      rewind 4
      pi=3.141592654
      conv=pi/180.
      read(1,*),freq
      ns=1
      write(2,7)
      write(2,8)
      write(5,1)
      write(5,8)
c     ******************************************************
c     *****   Read Phase and Compute 5 Second Averages   *****
c     ******************************************************
      j=0
      kt=20*ifix(freq)
      do 99 i=1,6
         read(4,*),phase
   99 continue
   10 j=j+1
      num(j)=0
      tot(j)=0.0
      tot2(j)=0.0
      do 20 i=1,kt
```

```
      read(4,*,end=25),phase
      t=phase*conv
      if (i.eq.1.and.j.eq.1) then
            t0=t
         else
            call correction(t,t0)
      endif
      tot(j)=tot(j)+t
      tot2(j)=tot2(j)+t*t
20 continue
   tot2(j)=tot2(j)/float(kt)
   tot(j)=tot(j)/float(kt)
   ds(j)=tot2(j)-tot(j)**2
   write(5,2)ds(j),tot(j)
   num(j)=j
   go to 10
c  ******************************************************************
c  *****   Identify Long Term Drifts, First Find Average Phase,   *****
c  *****   Ignore Records Less Than 5 Sec. in Duration            *****
c  ******************************************************************
25 j=j-1
26 call average(ds,avg,j)
   call standard(ds,avg,stan,j)
   var=stan * stan
   print*,var
   if (var.lt.0.1.or.j.lt.6) go to 28
   call search(tot,tot2,ds,num,j,stan,avg)
   call structure(tot,tot2,ds,j)
   go to 26
28 do 29 i=1,j
      write(2,2)ds(i),tot(i)
29 continue
   avg2=0.0
   avgx=0.0
   sum=0.0
   sig2=0.0
   avgy=0
   do 30 i=1,j
      avgy=avgy+tot(i)
      sum=sum+tot(i)*float(num(i))
      sig2=sig2+float(num(i)*num(i))
      avgx=avgx+float(num(i))
      avg2=avg2+(tot2(i)-(tot(i)**2))
30 continue
   avgx=avgx/float(j)
   avg2=avg2/float(j)
   avgy=avgy/float(j)
   sum=sum/float(j)
   sig2=sig2/float(j)-avgx*avgx
   slope=(sum-avgx*avgy)/sig2
```

```
      b=avgy-slope*avgx
      write(2,4)avgy,avg2
c     ************************************
c     *****   Now Correct for Drift   *****
c     ************************************
      do 40 i=1,j
         corr=slope*float(num(i))+b
         tot2(i)=tot2(i)-2.*corr*tot(i)+corr*corr
         tot(i)=tot(i)-corr
40    continue
c     *************************************
c     *****   Save Corrected Results   *****
c     *************************************
      write(3,3)
      write(3,8)
      avg1=0.0
      avg2=0.0
      avg3=0.0
      do 45 i=1,j
         ds(i)=tot2(i)-tot(i)**2
         avg3=avg3+ds(i)
         write(3,2)ds(i),tot(i)
         avg1=avg1+tot(i)
         avg2=avg2+tot2(i)
45    continue
      avg1=avg1/float(j)
      ds(j+1)=avg2/float(j)-avg1*avg1
      avg3=avg3/float(j)
      write(3,4)avg1,avg3
      write(3,9)ds(j+1)
      write(3,16)freq
      stop
      end
c
c
      subroutine search(mark,mark2,ds,num,j,stan,avg)
      real*4 mark(1000),mark2(1000),ds(1000)
      integer*4 num(1000)
      pi=3.141592654
      accept=avg + stan
      diff=0.
      i=1
10    if (i.le.j) then
         if (accept.lt.ds(i)) then
               do 20 k=i,j-1
                  mark(k)=mark(k+1)
                  mark2(k)=mark2(k+1)
                  num(k)=num(k+1)
                  ds(k)=ds(k+1)
20             continue
```

```
            j=j - 1
            diff=mark(i) - mark(1)
            i=i - 1
         else
            mark(i)=mark(i) - diff
            mark2(i)=ds(i) + mark(i) * mark(i)
      endif
      i=i + 1
      go to 10
   endif
   return
   end
c
c
   subroutine correction(phase,phi0)
   pi = 3.141592653589
10 flag=1.
   dphi=phase - phi0
   if (abs(dphi).gt.pi/2.) then
      if (dphi.eq.0.) then
            sign=1.
         else
            sign=dphi/abs(dphi)
      endif
      phase=phase - sign * pi / 2.
      flag=0.
   endif
   if (flag.eq.0) go to 10
   phi0=phase
   return
   end
c
c
   subroutine average(x,avg,num)
   dimension x(1000)
   sum=0.
   do 10 i=1,num
      sum=sum + x(i)
10 continue
   avg= sum / float(num)
   return
   end
c
c
   subroutine standard(x,avg,stan,num)
   dimension x(1000)
   diff=0.
   do 10 i=1,num
      diff=diff + (x(i) - avg) * (x(i) - avg)
10 continue
```

```
      diff = diff / float(num)
      stan=sqrt(diff)
      return
      end
c
c
      subroutine structure(mark,mark2,ds,j)
      real*4 mark(1000),mark2(1000),ds(1000)
      do 10 i= 1,j
         ds(i)=mark2(i) - mark(i) * mark(i)
   10 continue
      return
      end
```

```
c      ****************************************************
c      ****************************************************
c      **********              turbl                **********
c      **********   Finds the ampl tude structure   **********
c      **********      for a pair of channels.      **********
c      ****************************************************
c      ****************************************************
c
c
       program turbl
       open(1,file='/data/ampl.dat',status='old')
       open(2,file='/usr/data/aph')
       rewind 1
101    format(15x,'2048 Point Average = ',e13.6)
102    format(10x,'Amplitude Structure Function = ',e13.6)
       sum1=0.0
       sum2=0.0
       n=0
5      read(1,*,end=20),ampl1,ampl2
       if (abs(ampl1).lt.5.0.or.abs(ampl1).gt.5000.) go to 5
       if (abs(ampl2).lt.5.0.or.abs(ampl2).gt.5000.) go to 5
       n=n+1
       sum1=sum1+abs(ampl1)
       sum2=sum2+abs(ampl2)
       go to 5
20     if (n.ne.0) then
            avg1=sum1/float(n)
            avg2=sum2/float(n)
         else
            print*,"There are no acceptable amplitude values"
            go to 50
       endif
       rewind 1
       dx=0.0
       nn=0
25     sum1=0.0
       n=0
       do 30 i=1,2048
          read(1,*,end=40),ampl1,ampl2
          if (abs(ampl1).lt.5.0.or.abs(ampl1).gt.5000.) go to 30
          if (abs(ampl2).lt.5.0.or.abs(ampl2).gt.5000.) go to 30
          n=n+1
          ampl1=abs(ampl1/avg1)
          ampl2=abs(ampl2/avg2)
          ampl1=alog(ampl1)
          ampl2=alog(ampl2)
          sum1=sum1+(ampl1-ampl2)*(ampl1-ampl2)
30     continue
       if (n.ne.0) then
          dx=dx+sum1/float(n)
```

```
            sum1=sum1/float(n)
            write(2,101)sum1
            nn=nn+1
         endif
         go to 25
40       if (n.ne.0) then
               dx=(dx+sum1/float(n))/float(nn+1)
            else
               dx=dx/float(nn+1)
         endif
         write(2,102)dx
50       stop
         end
```

```
c      ******************************************************
c      *****                  Theory . f              *****
c      *****  Generate. data for the theoretical      *****
c      *****  plot of the structure function.         *****
c      ******************************************************
c
       program theory
       real mu2,L,nu,lambda,k
       character*64 xname,yname
       print*,"Name of x file"
       read*,xname
       print*,"Name of y file"
       read*,yname
       open(1,file = xname)
       open(2,file = yname)
       print*,"Propagation Distance in Meters"
       read*,r
       print*,"Mu**2"
       read*,mu2
       print*,"Outer Scale of Turblence in Meters"
       read*,L
       print*,"Temperature at run in Celsius"
       read*,T
       print*,"Frequency in Hz"
       read*,nu
       print*,"Input maximum separation"
       read*,sep
       ihigh = ifix(sep * 10.)
       ilow = 1
       pi = 3.141592654
       c0 = 331.6
       c = c0 * sqrt(1 + T / 273)
       lambda = c / nu
       k = 2 * pi / lambda
       k2 = k * k
       do 10 i = ilow, ihigh
          rho = float(i) / ( 10 * L )
          call integrate(rho,L,phi)
c-----------------------------------------------------------------c
c                                                                 c
c                          2       2                  phi         c
c      sf = sqrt(pi) * < u   > * k   * r * L * ( 1 - -----   )    c
c                                                     rho         c
c                                                                 c
c     where:                                                      c
c       sf      Structure Function                                c
c        2                                                        c
c      <u   >   Fluctuating Index of Refraction                   c
c        k      The Wave Number                                   c
c        r      Straight-Line Propagation Distance                c
```

```
c        L      Outer Scale of Turbulence                            c
c       phi     Integral from 0 to rho / L of exp(-x*x) dx           c
c       rho     Spatial Separation Perpendicular to the              c
c               Direction of Propagation                             c
c                                                                    c
c--------------------------------------------------------------------c
          sf = sqrt(pi) * mu2 * K2 * r * L * (1 - phi / rho)
          if (sf.le.0) go to 10
          write(1,*) rho * L
          write(2,*) sf
10    continue
      stop
      end
c
c---------------------------------------------------------------------c
c     Performs a Numerical Integration Using Trapazoid Rule          c
c     With Endpoint Correction                                       c
c---------------------------------------------------------------------c
c
      subroutine integrate(rho,L,phi)
      real L
      itter=20
      upper=rho
      lower=0
      dx=(upper - lower) / float(itter)
      x=lower
      call equation(x,y0)
      y=7 * y0
      call derivative(x,yp)
      y=y + dx * yp
      x=upper
      call equation(x,y0)
      y=y + 7 * y0
      call derivative(x,yp)
      y=y - dx * yp
      i=1
      x=lower + dx
10    if (x.lt.upper - dx / 2.) then
         if (2*ifix(i / 2).eq.i) then
               call equation(x,y0)
               y=y + 14 * y0
            else
               call equation(x,y0)
               y=y + 16 * y0
         endif
         x=x + dx
         i=i + 1
         go to 10
      endif
      phi= dx * y / 15.
```

```
      return
      end
c
c------------------------------------------------------------c
c     Performs a Numerical Derivative Using Difference      c
c     Tables.  This is Used in the Numerical Integration    c
c     Routine For Endpoint Correction                       c
c------------------------------------------------------------c
c
      subroutine derivative(x,yp)
      dimension delta(20),diff(20)
      dx = x / 100
      xx=x
      do 110 i=1,20
         call equation(xx,delta(i))
         xx=xx + dx
110   continue
      do 130 j=1,18
      do 120 i=1,20-j
         delta(i)=delta(i+1) - delta(i)
120   continue
      diff(j)=delta(1)
130   continue
      yp=0
      do 140 i=1,18
          if (dx.ne.0) then
               yp=yp + (-1.)**(i+1) * diff(i) / (dx * float(i))
            else
               yp=0
          endif
140   continue
      return
      end
c
c
      subroutine equation(x,y)
      y=exp(-x * x)
      return
      end
```

```
      program distrib
      character*64 fnam, xnam, ynam
      print*,"Root name for data"
      read*,fnam
      xnam = "x"//fnam
      ynam = "y"//fnam
      open(1,file=xnam,status="new")
      open(2,file=ynam,status="new")
      print*,"Insert p0"
      read*,p0
      print*,"Insert sigma"
      read*,sigma
      print*,"Upper limit for p"
      read*,pu
      print*,"Lower limit for p"
      read*,pl
      print*,"Increment for p"
      read*,pinc
      x0 = p0 / sigma
      xu = pu / sigma
      xl = pl / sigma
      xinc = pinc / sigma
      x = xl - xinc
10    x = x + xinc
      t = (x * x0) / 3.75
      if (t.ge.1) then
          call bivge1(x,x0,t,sigma,P)
        else
          call bivlt1(x,x0,t,sigma,P)
      endif
      write(1,*) x * sigma
      write(2,*) P
      if (x.lt.xu) go to 10
      stop
      end
c
c
      subroutine bivge1(x,x0,t,sigma,P)
      real I0
      I0=0.39894228+0.01328592/t+(0.002255319/(t*t))-(0.00157565/t**3)
     +      +(0.00916281/t**4)-(0.02057706/t**5)+(0.02635537/t**6)
     +      -(0.01647633/t**7)+(0.00392377/t**8)
      I0 = I0 / sqrt(x * x0)
      P = ((x * I0) / sigma) * exp(-((x - x0) * (x - x0)) / 2.)
      return
      end
c
c
      subroutine bivlt1(x,x0,t,sigma,P)
      real I0
```

```
      I0=1+3.5156229*t*t+3.0899424*t**4+1.2067492*t**6
     +     +0.2659732*t**8+0.0360768*t**10+0.0045813*t**12
      P = ((x * I0) / sigma) * exp(-(x * x + x0 * x0) / 2.)
      return
      end
```

```
****************************************************************
*
*  THIS PROGRAM IS DESIGNED TO FIT THE DATA USING
*  LEAST SQUARES TO A NORMALIZED BIVARIANT DISTRIBUTION.
*
****************************************************************

   program distfit
   parameter (maxd = 100, maxp = 5, npts = 500)
   character*1 yesno
   character*64 device
   real xdata(maxd), ydata(maxd), par(maxp)
   real xtheory(npts), ytheory(npts)
****************************************************************
*
*  ----- READ ALL THE DATA INTO ARRAYS AND KEEP COUNT -----
*
****************************************************************
   open (10,file='xdatafile')
   open (11,file='ydatafile')
   rewind 10
   rewind 11
****************************************************************
*
*  ----- EITHER ASK FOR GUESSES OR STORE THEM -----
*
****************************************************************
   print *,"best guess for p0 and sigma"
   read *, par(1), par(2)
   npar = 2
****************************************************************
*
*  ----- READ THE DATA POINTS -----
*
****************************************************************
   i = 1
10 read (10,*,end=20) xdata(i)
   read (11,*,end=20) ydata(i)
   i= i + 1
   goto 10
20 ndata = i - 1
   skip = 0.
****************************************************************
*
*  ----- NORMALIZE THE DATA -----
*
****************************************************************
   call integrate(xdata,ydata,ndata)
****************************************************************
*
```

```
*  ----- LOOP, GETTING BETTER PARAMETERS -----
*
**************************************************************
30 call curvefit (xdata,ydata,ndata,par,npar,sum)
**************************************************************
*
*  PLOT THE THEORETICAL CURVE AGAINST THE DATA
*  USING THE NEW PARAMETERS
*
**************************************************************
   if (skip.eq.0.) then
35   print *, "which device? screen, plotter, or none"
     read *,device
     if (device.ne."none") then
       if (device.ne."screen".and.device.ne."plotter") then
           print*,device,"is an invalid device"
           go to 35
       endif
**************************************************************
*
*      ---- CALCULATE THE THEORETICAL CURVE ----
*
**************************************************************
       call gtheory (xtheory,ytheory,par,npts)
       call plotdat (xdata,ydata,xtheory,ytheory,device,ndata,npts)
     endif
   endif
**************************************************************
*
*  ----- WRITE OUT NEW VALUES AND SAVE THEM -----
*
**************************************************************
   print *, "sum,     p0,      sigma"
   print *, sum, (par(i), i=1,npar)
   write (20,*) sum, (par(i), i=1,npar)
**************************************************************
*
*  ----- EITHER ASK NEW PARAMETERS OR USE OLD ONES -----
*
**************************************************************
   if (skip.eq.0.) then
       print *,"Do you wish to try again? (y/n/s)"
       read *, yesno
       if (yesno .eq. "s") then
         print*, "The number of iterations"
         read*, skip
         go to 30
       endif
       if (yesno .eq. "y") goto 30
     else
```

```
      skip = skip - 1.
      go to 30
   endif
   stop
   end

   subroutine curvefit(xdata,ydata,ndata,par,npar,sum)
****************************************************************
*
*           Fits theory to data by a least squares method
*           invented by Guass.
*           xdata,ydata    -   data points to fit(array)
*           ndata          -   number of points in array
*           par            -   array of quesses for parameters
*           npar           -   number of free parameters
*
****************************************************************
   parameter (maxp = 5)
   real xdata(*), ydata(*), par(*), q(maxp), u(maxp)
   real dyda(maxp), work1(maxp), work2(maxp), s(maxp*maxp)
   real theory
****************************************************************
*
*  ----- ZERO MAT S AND ARRAY Q -----
*
****************************************************************
   do 10 i = 1,maxp
10          q(i) = 0
   do 20 i = 1, maxp*maxp
20          s(i) = 0
****************************************************************
*
*  ----- STEP THROUGH ALL OF THE DATA POINTS -----
*
****************************************************************
   weight = 0.
   sum = 0.
   do 30 i = 1,ndata
            xvalue = xdata(i)
            ytheo = theory(par,xvalue)
            sum = sum + (ytheo - ydata(i)) ** 2
            call deriv(par,xvalue,ytheo,dyda)
            do 30 k = 1, npar
                     q(k) = q(k) + dyda(k) * (ytheo - ydata(i))
                     weight = weight + dyda(k) * ytheo
                              do 30 m=1,npar
*                             *******************************
*                             ---- INDEX ON S LIKE 2 DIMENSION
*                             *******************************
                              km = npar * (m - 1) + k
```

```
                        s(km) = s(km) + dyda(k) * dyda(m)
30 continue
   weight = 2. * sum / weight
   do 35 k = 1, npar
           kk = npar * (k - 1) + k
           s(kk) = s(kk) + 1. / weight
35 continue
   call gminv (s,npar,det,work1,work2)
   call gmprd (s,q,u,npar,npar,1)
*****************************************************************
*
*  ----- VECTOR U NOW CONTAINS PARAMETER ADJUSTMENT -----
*  ----- ADD THEM TO VECTOR "PAR" -----
*
*****************************************************************
   do 40 i = 1,npar
           par(i) = par(i) - u(i)
40 continue
   sum = 0
   do 50 i= 1,ndata
           ytheo = theory(par,xvalue)
           sum = sum + (ydata(i) - ytheo) ** 2
50 continue
   return
   end

   real function theory(par,xvalue)
*****************************************************************
*
*  THIS FUNCTION CALCULATES A YVALUE FOR THE
*  DISTRIBUTION
*
*****************************************************************
   real par(*)
   sigma = par(2)
   x0 = par(1) / sigma
   x = xvalue / sigma
   t = (x * x0) / 3.75
   if (t.ge.1) then
      call bivge1(x,x0,t,sigma,P)
     else
      call bivlt1(x,x0,t,sigma,P)
   endif
   theory = P
   return
   end
c
c
   subroutine bivge1(x,x0,t,sigma,P)
   real I
```

```
      I=0.39894228+0.01328592/t+(0.002255319/(t*t))-(0.00157565/t**3)
     +        +(0.00916281/t**4)-(0.02057706/t**5)+(0.02635537/t**6)
     +        -(0.01647633/t**7)+(0.00392377/t**8)
      I = I / sqrt(x * x0)
      P = ((x * I) / sigma) * exp(-((x - x0) * (x - x0)) / 2.)
      return
      end
c
c
      subroutine bivlt1(x,x0,t,sigma,P)
      real I
      I=1+3.5156229*t*t+3.0899424*t**4+1.2067492*t**6
     +       +0.2659732*t**8+0.0360768*t**10+0.0045813*t**12
      P = ((x * I) / sigma) * exp(-(x * x + x0 * x0) / 2.)
      return
      end

      subroutine deriv(par,xvalue,ytheo,dyda)
*****************************************************************
*
*  EVALUATE THE DERIVATIVES (DYDA) OF THE THE THEORY
*  WITH RESEPCT TO THE ADJUSTMENT PARAMETERS.
*
*****************************************************************
      real par(*), dyda(*), I1, I0, Iratio
      sigma = par(2)
      x = xvalue / sigma
      x0 = par(1) / sigma
      z = x * x0
      Iratio = I1(z) / I0(z)

      dyda(1) = ytheo / sigma * (-x0 + x * Iratio)
      dyda(2) = 1. - (x*x+x0*x0)/2. + z * Iratio
      dyda(2) = -2. * ytheo / sigma * dyda(2)

      return
      end

      real function I0(z)
      real I0
      t = z / 3.75
      if (t.ge.1) then
          poly=0.39894228+0.01328592/t+0.002255319/(t*t)
     +         -0.00157565/t**3+0.00916281/t**4-0.02057706/t**5
     +         +0.02635537/t**6-0.01647633/t**7+0.00392377/t**8
         I0 = poly
        else
         I0 = 1.+3.5156229*t*t+3.0899424*t**4+1.2067492*t**6
     +            + 0.2659732*t**8+0.0360768*t**10+0.0045813*t**12
      endif
```

```
      return
      end

      real function I1(z)
      real I1
      t = z / 3.75
      if (t.ge.1) then
         poly=0.39894228-0.03988024/t-0.00362018/(t*t)
     +         +0.00163801/t**3-0.01031555/t**4+0.02282967/t**5
     +         -0.02895312/t**6+0.01787654/t**7-0.00420059/t**8
         I1 = poly
        else
         poly=0.5+0.87890594*t*t+0.51498869*t**4+0.15084934*t**6
     +         +0.02658733*t**8+0.00301532*t**10+0.00032411*t**12
         I1 = poly * z
      endif
      return
      end

      subroutine gminv(a,n,d,l,m)
*****************************************************************
*
*         SUBROUTINE MINV
*
*         PURPOSE
*            INVERT A MATRIX
*
*         USAGE
*            CALL MINV(A,N,D,L,M)
*
*         DESCRIPTION OF PARAMETERS
*            A - INPUT MATRIX, DESTROYED IN COMPUTATION AND REPLACED BY
*                RESULTANT INVERSE.
*            N - ORDER OF MATRIX A
*            D - RESULTANT DETERMINANT
*            L - WORK VECTOR OF LENGTH N
*            M - WORK VECTOR OF LENGTH N
*
*         REMARKS
*            MATRIX A MUST BE A GENERAL MATRIX
*
*         SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
*            NONE
*
*         METHOD
*            THE STANDARD GAUSS-JORDAN METHOD IS USED. THE DETERMINANT
*            IS ALSO CALCULATED. A DETERMINANT OF ZERO INDICATES THAT
*            THE MATRIX IS SINGULAR.
*
*****************************************************************
```

```
      dimension a(1),l(1),m(1)
****************************************************************
*
*           IF A DOUBLE PRECISION VERSION OF THIS ROUTINE IS DESIRED, THE
*           C IN COLUMN 1 SHOULD BE REMOVED FROM THE DOUBLE PRECISION
*           STATEMENT WHICH FOLLOWS.
*
*        double precision a,d,biga,hold,dabs
*
*           THE C MUST ALSO BE REMOVED FROM DOUBLE PRECISION STATEMENTS
*           APPEARING IN OTHER ROUTINES USED IN CONJUNCTION WITH THIS
*           ROUTINE.
*
*           THE DOUBLE PRECISION VERSION OF THIS SUBROUTINE MUST ALSO
*           CONTAIN DOUBLE PRECISION FORTRAN FUNCTIONS.  ABS IN STATEMENT
*           10 MUST BE CHANGED TO DABS.
*
****************************************************************
*
*           SEARCH FOR LARGEST ELEMENT
*
****************************************************************
      d=1.0
      nk=-n
      do 80 k=1,n
              nk=nk+n
              l(k)=k
              m(k)=k
              kk=nk+k
              biga=a(kk)
              do 20 j=k,n
                      iz=n*(j-1)
                      do 20 i=k,n
                              ij=iz+i
   10                                 if( abs(biga)- abs(a(ij))) 15,20,20
   15                                 biga=a(ij)
                              l(k)=i
                              m(k)=j
   20                 continue
****************************************************************
*
*             ---- INTERCHANGE ROWS ----
*
****************************************************************
              j=l(k)
              if(j-k) 35,35,25
   25                 ki=k-n
              do 30 i=1,n
                      ki=ki+n
                      hold=-a(ki)
```

```
                    ji=ki-k+j
                    a(ki)=a(ji)
   30                         a(ji) =hold
*****************************************************************
*
*          ---- INTERCHANGE COLUMNS ----
*
*****************************************************************
   35               i=m(k)
          if(i-k) 45,45,38
   38               jp=n*(i-1)
          do 40 j=1,n
                    jk=nk+j
                    ji=jp+j
                    hold=-a(jk)
                    a(jk)=a(ji)
   40                         a(ji) =hold
*****************************************************************
*
*         DIVIDE COLUMN BY MINUS PIVOT (VALUF OF PIVOT
*         ELEMENT IS CONTAINED IN BIGA)
*
*****************************************************************
   45               if(biga) 48,46,48
   46               d=0.0
          return
   48               do 55 i=1,n
                    if(i-k) 50,55,50
   50                         ik=nk+i
                    a(ik)=a(ik)/(-biga)
   55               continue
*****************************************************************
*
*          ---- REDUCE MATRIX ----
*
*****************************************************************
          do 65 i=1,n
                    ik=nk+i
                    hold=a(ik)
                    ij=i-n
                    do 65 j=1,n
                              ij=ij+n
                              if(i-k) 60,65,60
   60                                   if(j-k) 62,65,62
   62                                   kj=ij-i+k
                              a(ij)=hold*a(kj)+a(ij)
   65               continue
*****************************************************************
*
*          ---- DIVIDE ROW BY PIVOT ----
```

```
*
*****************************************************************
          kj=k-n
          do 75 j=1,n
                  kj=kj+n
                  if(j-k) 70,75,70
  70                      a(kj)=a(kj)/biga
  75              continue
*****************************************************************
*
*         ---- PRODUCT OF PIVOTS ----
*
*****************************************************************
          d=d*biga
*****************************************************************
*
*         ---- REPLACE PIVOT BY RECIPROCAL ----
*
*****************************************************************
          a(kk)=1.0/biga
  80      continue
*****************************************************************
*
*       ---- FINAL ROW AND COLUMN INTERCHANGE ----
*
*****************************************************************
  k=n
  100     k=(k-1)
   if(k) 150,150,105
  105     i=l(k)
   if(i-k) 120,120,108
  108     jq=n*(k-1)
   jr=n*(i-1)
   do 110 j=1,n
          jk=jq+j
          hold=a(jk)
          ji=jr+j
          a(jk)=-a(ji)
  110             a(ji) =hold
  120     j=m(k)
   if(j-k) 100,100,125
  125     ki=k-n
   do 130 i=1,n
          ki=ki+n
          hold=a(ki)
          ji=ki-k+j
          a(ki)=-a(ji)
  130             a(ji) =hold
   go to 100
  150     return
```

```
      end

      subroutine gmprd(a,b,r,n,m,l)
*************************************************************
*
*        SUBROUTINE GMPRD
*
*        PURPOSE
*           MULTIPLY TWO GENERAL MATRICES TO FORM A RESULTANT GENERAL
*           MATRIX
*
*        USAGE
*           CALL GMPRD(A,B,R,N,M,L)
*
*        DESCRIPTION OF PARAMETERS
*           A - NAME OF FIRST INPUT MATRIX
*           B - NAME OF SECOND INPUT MATRIX
*           R - NAME OF OUTPUT MATRIX
*           N - NUMBER OF ROWS IN A
*           M - NUMBER OF COLUMNS IN A AND ROWS IN B
*           L - NUMBER OF COLUMNS IN B
*
*        REMARKS
*           ALL MATRICES MUST BE STORED AS GENERAL MATRICES
*           MATRIX R CANNOT BE IN THE SAME LOCATION AS MATRIX A
*           MATRIX R CANNOT BE IN THE SAME LOCATION AS MATRIX B
*           NUMBER OF COLUMNS OF MATRIX A MUST BE EQUAL TO NUMBER OF ROW
*           OF MATRIX B
*
*        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
*           NONE
*
*        METHOD
*           THE M BY L MATRIX B IS PREMULTIPLIED BY THE N BY M MATRIX A
*           AND THE RESULT IS STORED IN THE N BY L MATRIX R.
*
*************************************************************
      dimension a(1),b(1),r(1)
      ir=0
      ik=-m
      do 10 k=1,l
            ik=ik+m
            do 10 j=1,n
                  ir=ir+1
                  ji=j-n
                  ib=ik
                  r(ir)=0
                  do 10 i=1,m
                        ji=ji+n
```

```
                         ib=ib+1
10                                   r(ir)=r(ir)+a(ji)*b(ib)
      return
      end


      subroutine plotdat(xm,ym,xt,yt,device,ndata,npts)
************************************************************
*
*     Subroutine to Plot Measured Data With Theory
*
************************************************************
      include "/usr/include/libmp.f"
      integer*4 gls(sizeofgca)
      character*64 device
      real xm(*),ym(*),xt(*),yt(*)
      integer bun(2), lin(2)
      bun(1) = 0
      bun(2) = 66
      lin(1) = 1
      lin(2) = 0
      if (device.eq."screen") call system("cleargp")
      call mpinit(gls)
      call mplotsrcx(gls,1,ndata,0,xm,"F",1,1,null,null)
      call mplotsrcy(gls,1,ndata,0,ym,"F",1,1,null,null)
      call mplotsrcx(gls,2,npts,0,xt,"F",1,1,null,null)
      call mplotsrcy(gls,2,npts,0,yt,"F",1,1,null,null)
      call mplotchrs(gls,"*",1,nullintray,nullintray)
      call mplines(gls,2,bun,lin)
      if (device.eq."screen") then
         call mpdevice(gls,"mcd",2,0)
       else
         call mpdevice(gls,"hpgl0",2,0)
      endif
      call mplot(gls,0,0,0)
      call mpend(gls)
      read*,wait
      if (device.eq."screen") call system("cleargp")
      return
      end


      subroutine gtheory (xt,yt,par,npts)
************************************************************
*
*     ---- CALCULATES THE THEORETICAL CURVE ----
*
************************************************************
      real xt(*), yt(*), lolim, uplim, par(*), incr, diff
      real theory
```

```
      print *, "upper limit"
      read *, uplim
      print *, "lower limit"
      read *, lolim
      diff = uplim - lolim
      incr = diff / (npts - 1)
      xt(1) = lolim
      yt(1) = theory(par,xt(1))
      do 100 i = 2, npts
              xt(i) = xt(i-1) + incr
              yt(i) = theory (par,xt(i))
100           continue
      return
      end


      subroutine integrate(xdata,ydata,ndata)
*****************************************************************
*
*   INTEGRATES THE DATA USING THE TRAPOZID METHOD
*
*****************************************************************
      real xdata(*), ydata(*)
      area = 0.
      do 10 i = 1, ndata - 1
              dx = xdata(i+1) - xdata(i)
              avghi = (ydata(i+1) + ydata(i)) / 2.
              area = area + dx * avghi
10    continue
*****************************************************************
*
*   ---- NORMALIZE THE DATA NOW ----
*
*****************************************************************
      do 20 i = 1, ndata
              ydata(i) = ydata(i) / area
20    continue
      return
      end
```

```
      program stordat
      character*64 root,xroot, yroot
      print*,"Input root for data file"
      read*,root
      xroot = "x"//root
      yroot = "y"//root
      open(1,file=xroot,status="new")
      open(2,file=yroot,status="new")
      print*,"Input number of points"
      read*,npoints
      print*,"Input position of peak in dB"
      read*,pdb
      print*,"Input position of peak from scale"
      read*,pscale
      print*,"Input height of peak"
      read*,height
      print*,"Input position of reference peak in dB"
      read*,prefdb
      scale = 1. / height
      Pref = 10**(prefdb / 20.)
      do 30 i = 1, npoints
        print*,"Input x",i,",y",i
        read*,p,prob
        xdb = pdb + 20. * log10(p / pscale)
        x = 10**(xdb / 20.) / Pref
        y = prob * scale
        write(1,*) x
        write(2,*) y
30    continue
      stop
      end
```

```
c-----------------------------------------------------------c
c     This program creates the data for constructing        c
c     a scatter plot                                        c
c-----------------------------------------------------------c
c
      program scatter
      open(1,file="phase.dat")
      open(2,file="ampl.dat")
      open(3,file="xfile")
      open(4,file="yfile")
      open(20,file="limits")
      rewind 1
      rewind 2
      xmax=-100000
      xmin=100000
      ymax=-100000
      ymin=100000
      conv = 3.14159/180.
10    read(1,*,end=20) phase
      read(2,*,end=20) amp1,amp2
      if (amp1.ne.0.and.amp2.ne.0.) then
        amp = abs(amp1 / amp2)
c-----------------------------------------------c
c     Convert the data from rectangular        c
c     to polar coordinates                     c
c-----------------------------------------------c
        x = amp * cos(phase * conv)
        y = amp * sin(phase * conv)
c-----------------------------------------------c
c     Find the range of the data               c
c-----------------------------------------------c
        xmax = amax1(x,xmax)
        ymax = amax1(y,ymax)
        xmin = amin1(x,xmin)
        ymin = amin1(y,ymin)
        write(3,*) x
        write(4,*) y
      endif
      go to 10
20    xmax = amax1(abs(xmax),abs(xmin))
      ymax = amax1(abs(ymax),abs(ymin))
      xmin = - xmax
      ymin = - ymax
      write(20,*) xmax,xmin,ymax,ymin
      stop
      end
```

```
c       ***************************************************************
c       ***************************************************************
c       **********            nuphazscat.f                  **********
c       **********   Finds the amplitude and phase          **********
c       **********      difference between two              **********
c       **********        different channels.               **********
c       **********   For use in the scatter program         **********
c       ***************************************************************
c       ***************************************************************
c
c
        program nuphazscat

        integer hiwin, i, iblk, ch1, ch2, dgfreq, ipts, j
        integer k, lowin, n1, n2, xcycl1, xcycl2, nk
        integer npass, numch
        integer*2   data(1024,2)
        real    area1, area2, bias1, bias2
        real    data1(1024), data2(1024), lastp1, lastp2
        real    noise1, noise2, phasinc, phastot, pi
        real    period, sumtim, tsampl, timkp1, timkp2
        real    time(1024,2), phase(2048), ampl(1024,2)
        parameter (pi=3.141592654)
        open(1,file='holder',status='old')
        open(2,file='ampl.dat')
        open(4,file='phase.dat')
        open(20,file='data',access='direct',recl=2)
        rewind 1
        read(1,*),numch
        read(1,*),dgfreq
        read(1,*),ipts
        read(1,*),ch1
        read(1,*),ch2

        tsampl = 1 / float(dgfreq)
        npass = ipts / (numch * 1024)
        xcycl1 = 0
        xcycl2 = 0
        lastp1 = 0.
        lastp2 = 0.
        area1 = 0.
        area2 = 0.
        phastot = 0.
        iblk = 0
        nk = 0
c       ******************************************
c       ** Read In 1024 Points From Each Channel **
c       ** Skip First Hundred Points             **
c       ******************************************
10      lowin = ch1 + nk * numch + 100 * numch
```

```
      hiwin = ch1 + (nK + 1023) * numch + 100 * numch
      j = 1
      do 2 i=lowin,hiwin,numch
         read(20,rec=i),data(j,1)
         read(20,rec=i+ch2-ch1),data(j,2)
         j = j + 1
 2    continue
      nK = nK + 1024
c     ***************************************************
c     *****  Determine DC Bias  (Just Average Over   *****
c     *****  Complete Cycles, 1020 instead of 1024)  *****
c     ***************************************************
      bias1 = 0.0
      bias2 = 0.0
      do 15 i=1,1020
         bias1 = bias1 + float(data(i,1))
         bias2 = bias2 + float(data(i,2))
15    continue
      bias1 = bias1 / 1020.
      bias2 = bias2 / 1020.
c     ********************
c     ** Remove DC Bias **
c     ********************
      do 20 i = 1, 1024
         data1(i) = data(i,1) - bias1
         data2(i) = data(i,2) - bias2
20    continue
c
      n1=xcycl1+1
      n2=xcycl2+1
c     *****************************************************
c     *****  Determine Local Zeroes For Channels 1 & 2  *****
c     *****************************************************
      call cycle(1,data1,n1,lastp1,timKp1,area1,time,ampl,
     +                  tsampl)
      n1=n1-1
      call cycle(2,data2,n2,lastp2,timKp2,area2,time,ampl,
     +                  tsampl)
      n2=n2-1
c     **********************************************************
c     *****   Zero Crossings are now in       *****
c     *****   Time Array.  Compute Period    *****
c     *******************************************
      sumtim = 0.0
      do 65 i = 1, n1
              sumtim = sumtim + time(i,1)
65    continue
      period = 2. * sumtim / float(n1)
c     **********************************************************
c     *****   Check Signal to Noise Ratio for this Block   *****
```

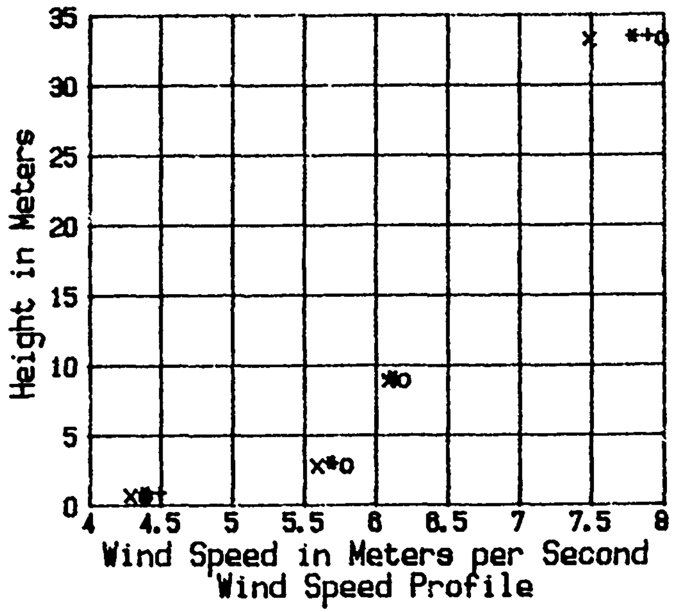
```
c     ******************************************
      noise1=float(nzer1)/float(n1)*100.
      noise2=float(nzer2)/float(n2)*100.
      if (noise1.lt.5.) then
         if (noise2.lt.5.) then
c           ****************************
c           *****   Compute Phase   *****
c           ****************************
              phasinc=(time(1,1)-time(1,2))/period*360.
              phase(1)=phastot+phasinc
              k=min0(n1,n2)
              do 70 i=2,k
                 phasinc=(time(i,1)-time(i,2))/period*360.
                 phase(i)=phase(i-1)+phasinc
70            continue
              do 75 i=1,k
                 if (iblk.eq.0.and.i.eq.1) go to 75
                 write(2,*)ampl(i,1),ampl(i,2)
                 write(4,*)phase(i)
75            continue
              phastot=phase(k)
           else
              print*,'Ch 2. Too Noisy. Skip Data. Hold Phase.'
         endif
      else
         print*,'Ch1. Too Noisy. Skip Data. Hold Phase.'
      endif
      iblk=iblk+1
      if (iblk.ge.npass) go to 95
      xcycl1 = 0
      xcycl2 = 0
      if (n2.gt.n1) then
c        *********************************
c        *****   More Cycles on Ch 2  *****
c        *********************************
         xcycl2=n2-n1
         do 85 i=1,xcycl2
            time(i,2)=time(n1+i,2)
            ampl(i,2)=ampl(n1+i,2)
85       continue
      endif
      if (n1.gt.n2) then
c        *********************************
c        *****   More Cycles on Ch 1  *****
c        *********************************
         xcycl1=n1-n2
         do 90 i=1,xcycl1
            time(i,1)=time(n2+i,1)
            ampl(i,1)=ampl(n2+i,1)
90       continue
```

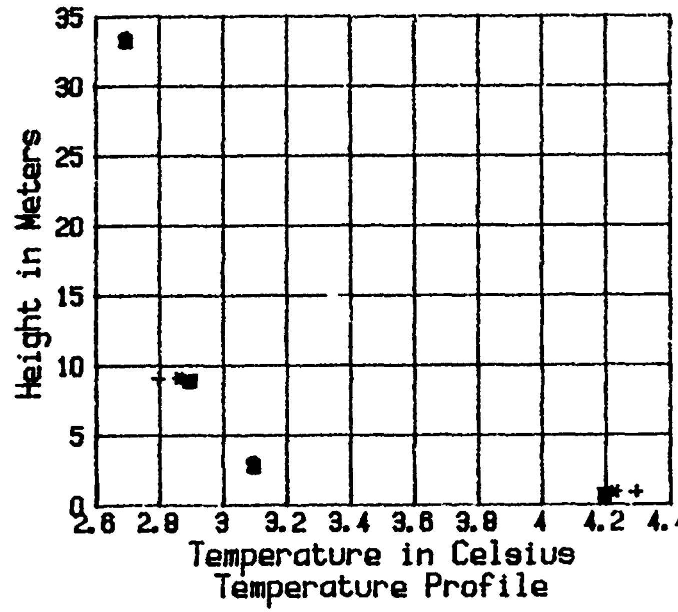
```
      endif
      go to 10
95    stop
      end
c
c
c     ***********************************
c     ** Finds the Zero Crossing Between **
c     **      (0,y1) And (dx,y2)         **
c     ***********************************
      real function zerox(y1,y2,dx)
      real     y1, y2, slope, dx
      slope = (y2 - y1) / dx
      zerox = - y1 / slope
      return
      end
c
c
      subroutine cycle(nch,data,ncyc,lastpt,timekp,area,
     +                 time, ampl, tsampl)
c     *****************************************************
c     **  Processes Cycles in Each Block of 1024 Points   **
c     **  Records Time and Amplitude of each 1/2 cycle    **
c     *****************************************************
c
      integer   i, ncyc, nch
      real      ampl(1024,2), time(1024,2), data(1024)
      real      aftime, zerox, thispt, lastpt, prod, timekp, tsampl
      real      pi, area
      parameter (pi=3.141592654)
c
      do 10 i=1,1024
         thispt=float(data(i))
         prod = thispt*lastpt
         if (prod.ge.0) then
c           ******************************
c           *****  Same Side of Zero  *****
c           ******************************
            timekp = timekp + tsampl
            area = area + tsampl*(thispt+lastpt)/2
            if (thispt.eq.0) then
c              ******************************
c              *****  Hit Zero Exactly  *****
c              ******************************
               time(ncyc,nch) = timekp
               ampl(ncyc,nch) = (pi*area)/(2*timekp)
               ncyc = ncyc + 1
               timekp = 0
               area = 0
            endif
```

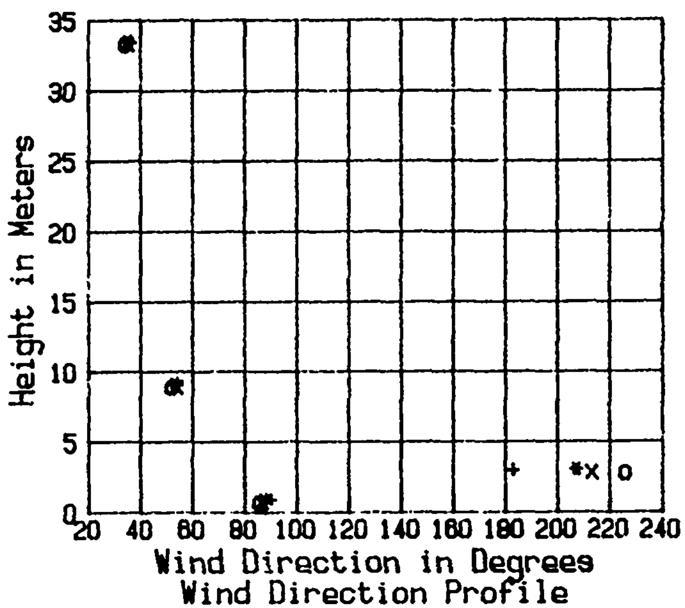
```
          else
c             *****************************************
c             *****  Crossed Zero -- End of 1/2 Cycle  *****
c             *****************************************
              aftime = zerox(lastpt,thispt,tsampl)
              timeKp = timeKp + aftime
              area = area + lastpt*aftime/2
              time(ncyc,nch) = timeKp
              ampl(ncyc,nch) = (pi*area)/(2*timeKp)
              ncyc = ncyc + 1
c             *****************************************
c             *****  Get Prepared For Next 1/2 Cycle  *****
c             *****************************************
              timeKp = tsampl - aftime
              area = thispt*timeKp/2
          endif
c         *************************************
c         *****  Remember This Data Point  *****
c         *************************************
          lastpt = thispt
10      continue
c
        return
        end
```

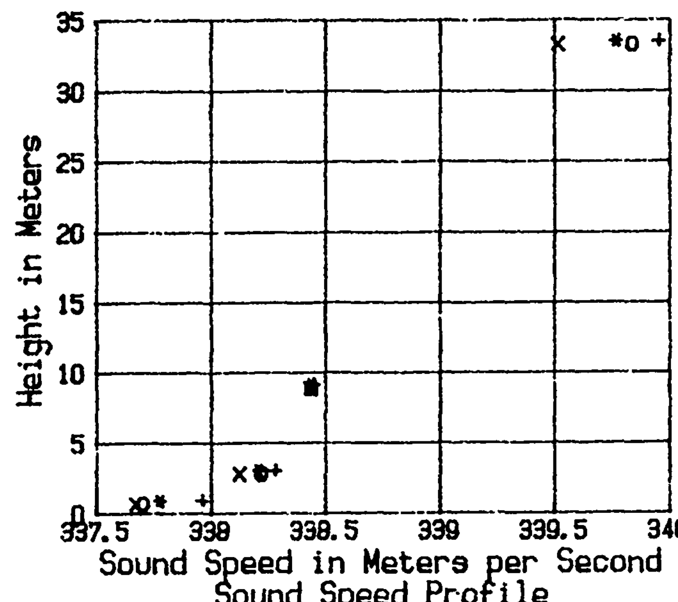
APPENDIX C

Plots of the meteorological parameters measured during the experiment.

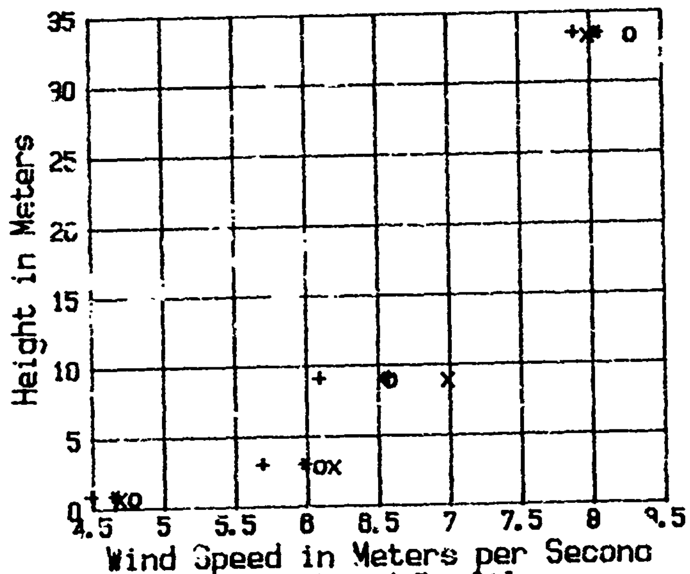




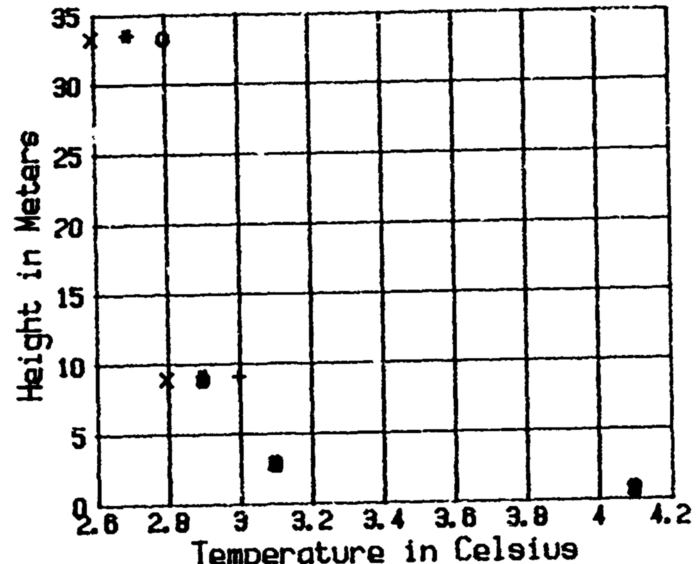




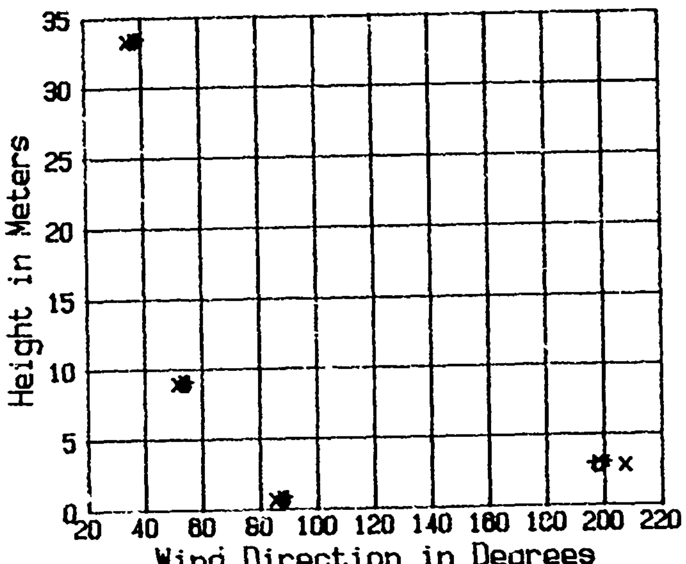
December 13, 1984
Run 1.1



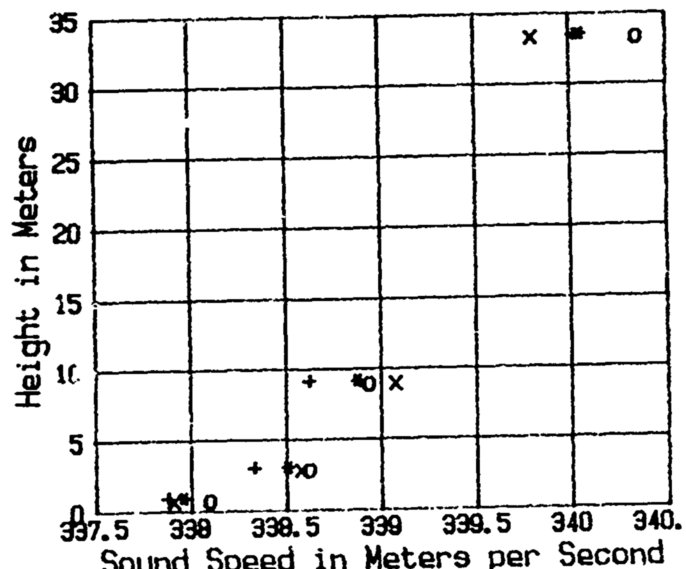
Wind Speed Profile



Temperature Profile

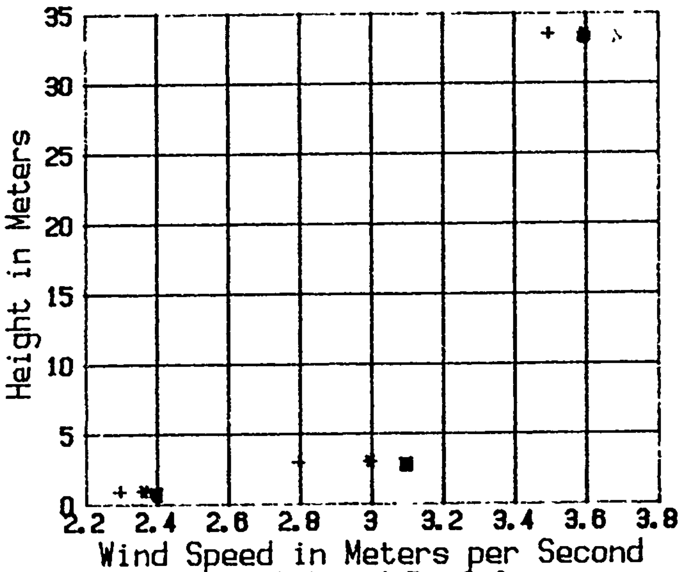


Wind Direction Profile

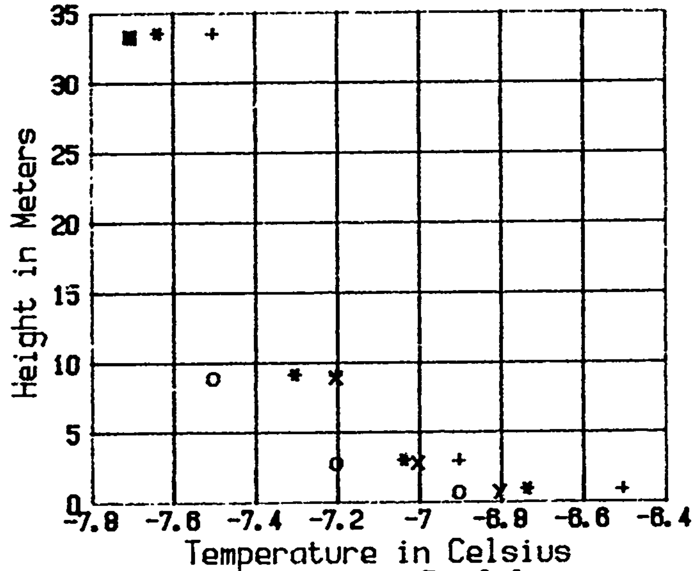


Sound Speed Profile

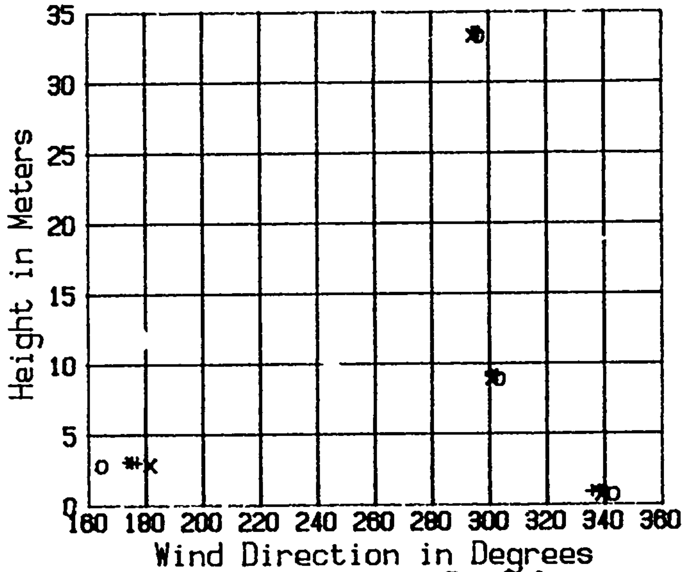
December 13, 1984
Run 1.2



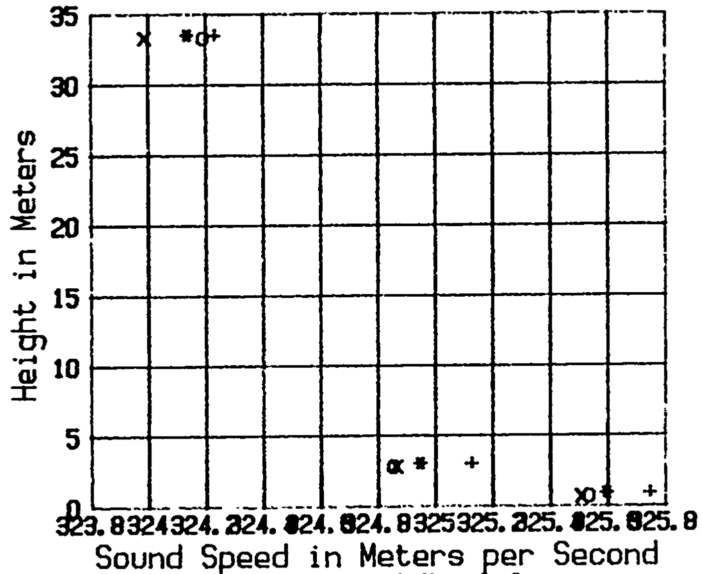
Wind Speed Profile



Temperature Profile

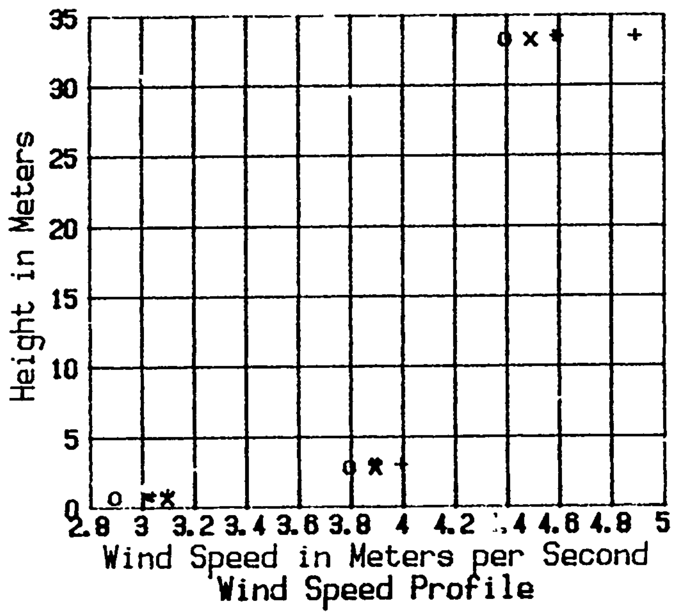


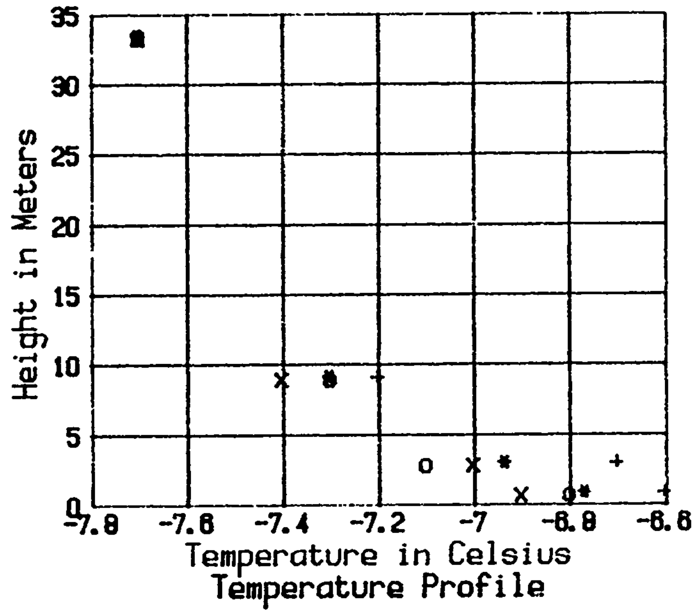
Wind Direction Profile

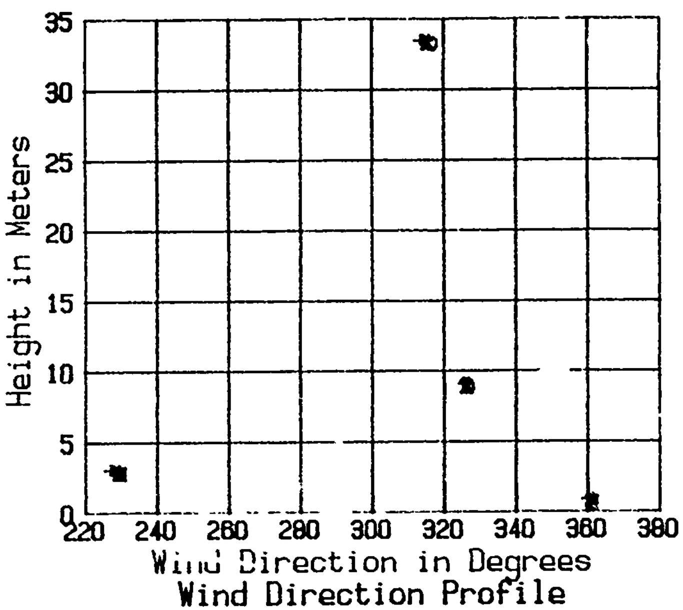


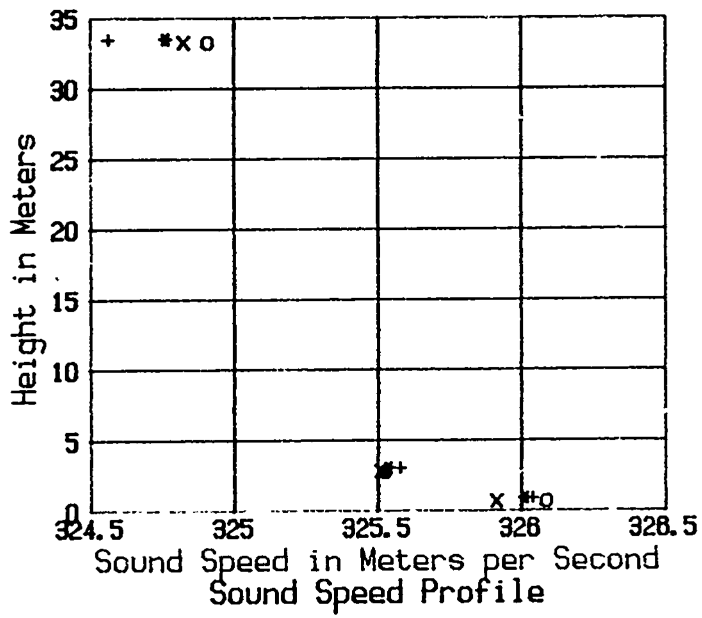
Sound Speed Profile

January 11, 1985
Run 2.1
Bondville, IL

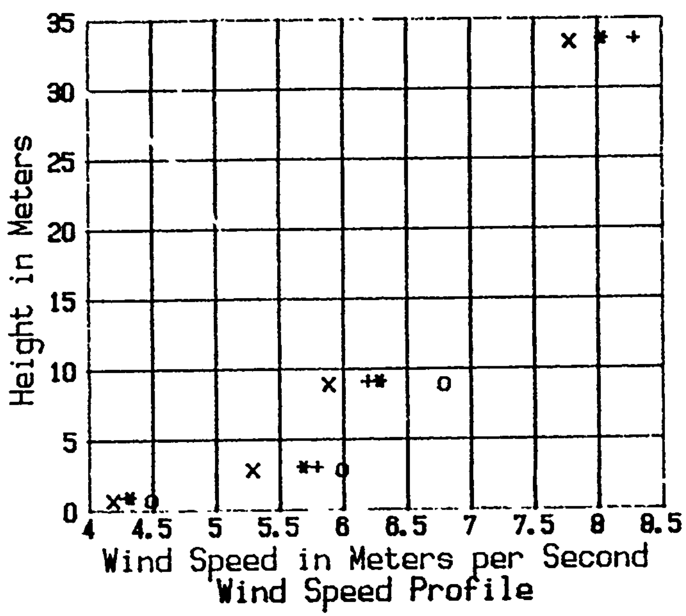


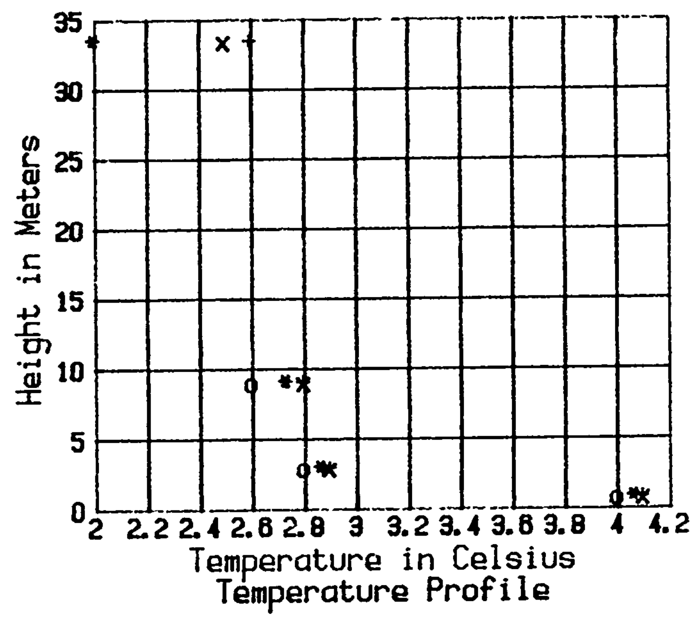


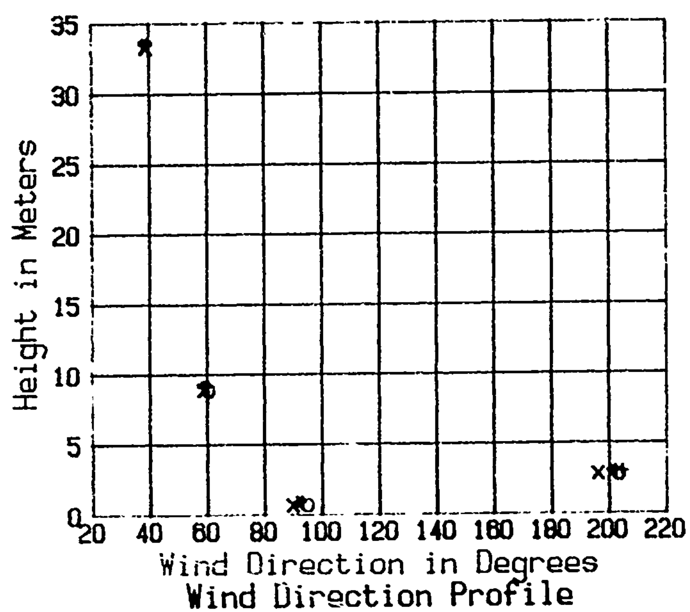


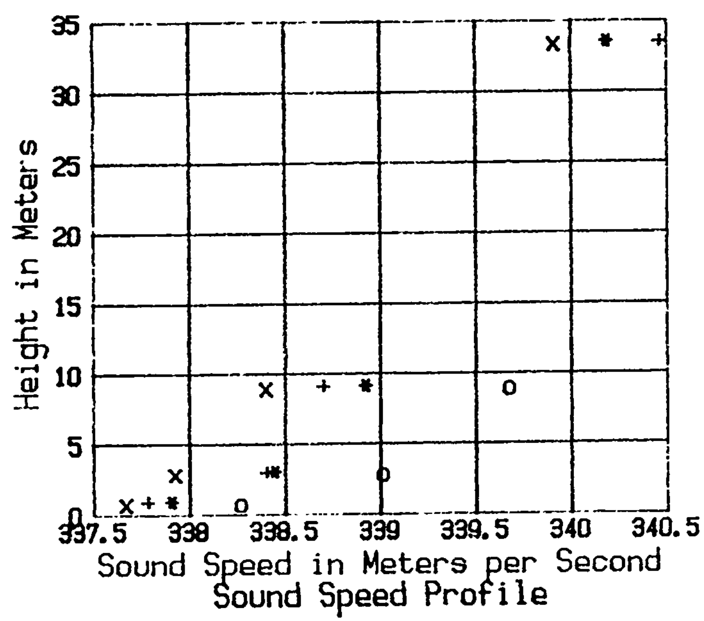


January 11, 1985
Run 2.2
Bondville, IL

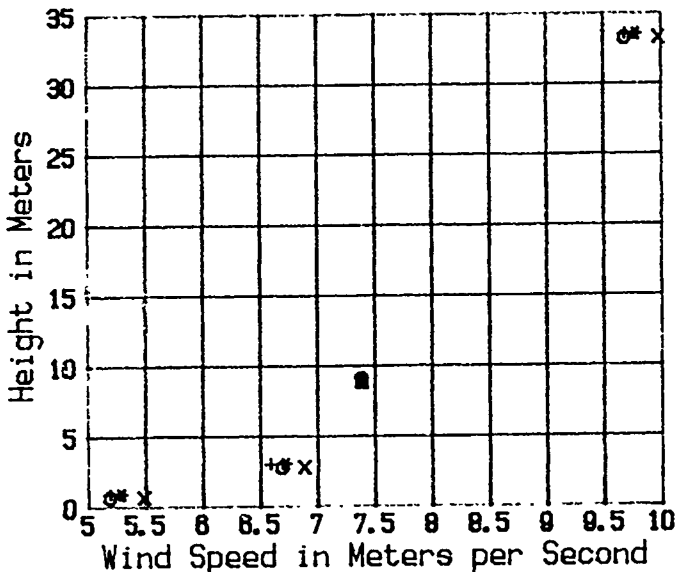




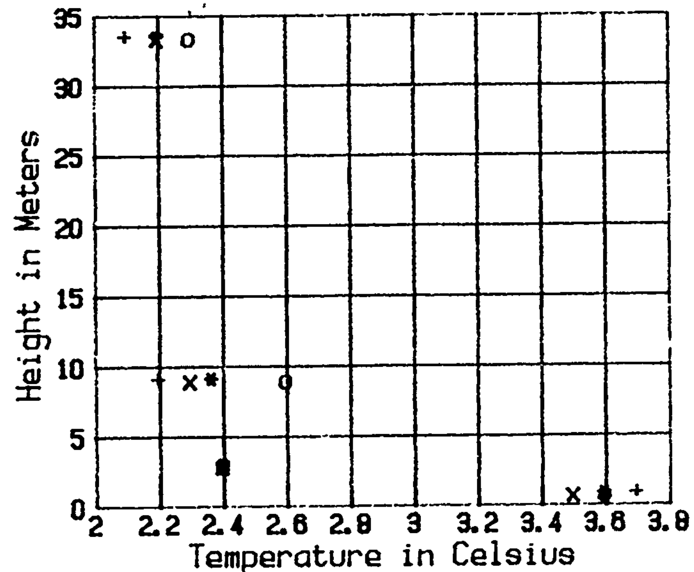




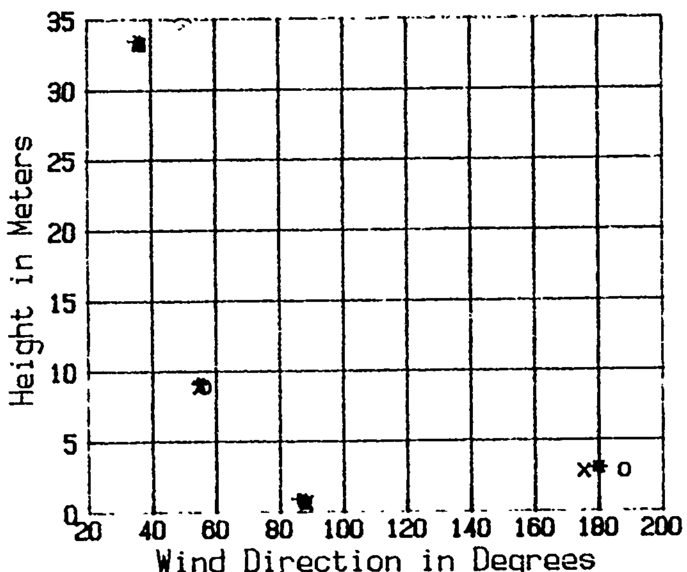
December 13, 1984
Run 2.1
Bondville, IL



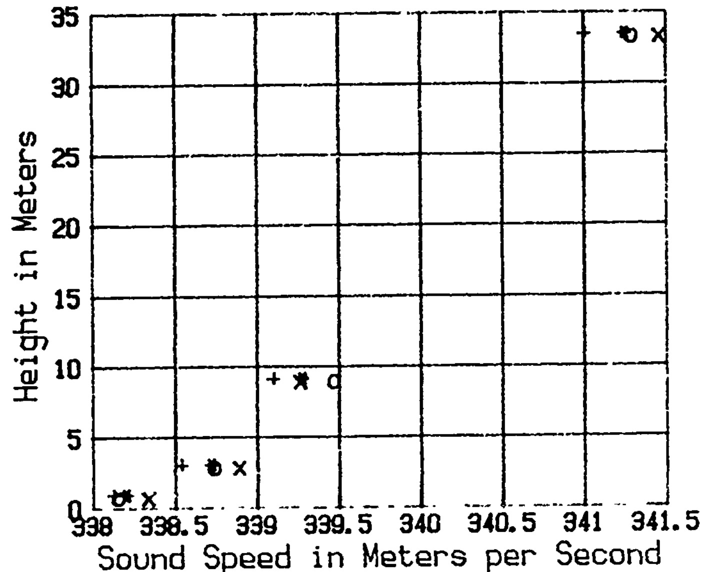
Wind Speed Profile



Temperature Profile

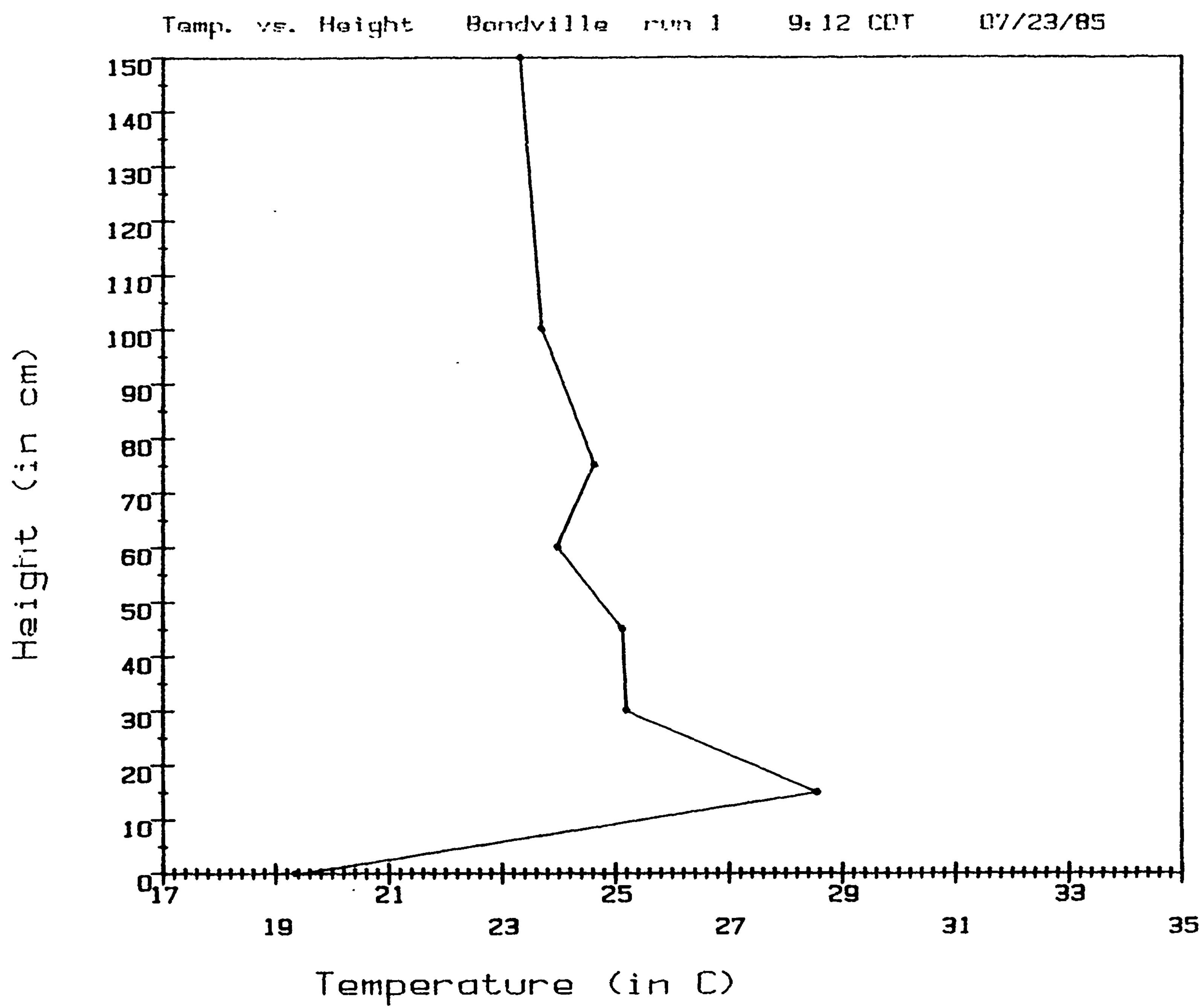


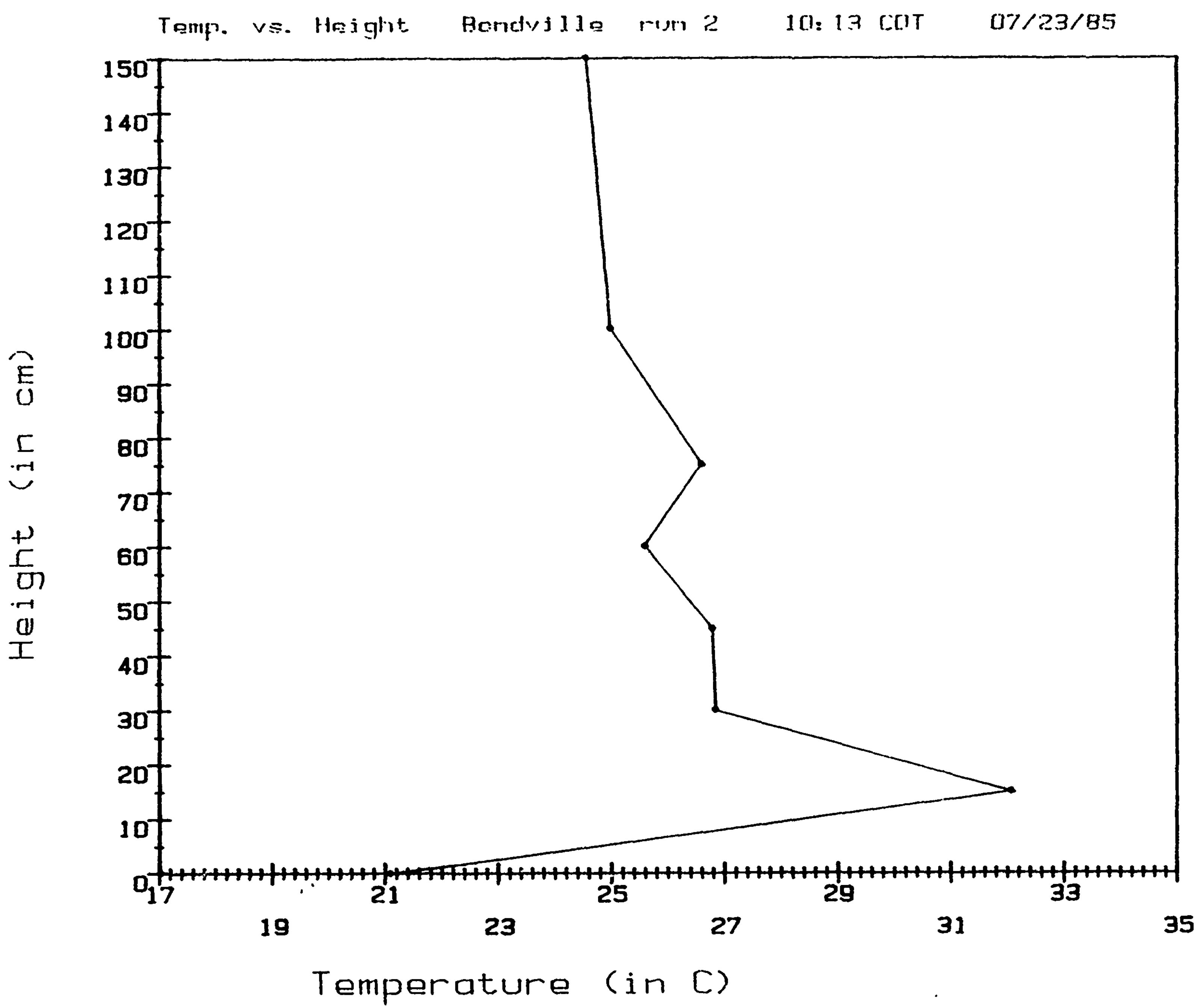
Wind Direction Profile

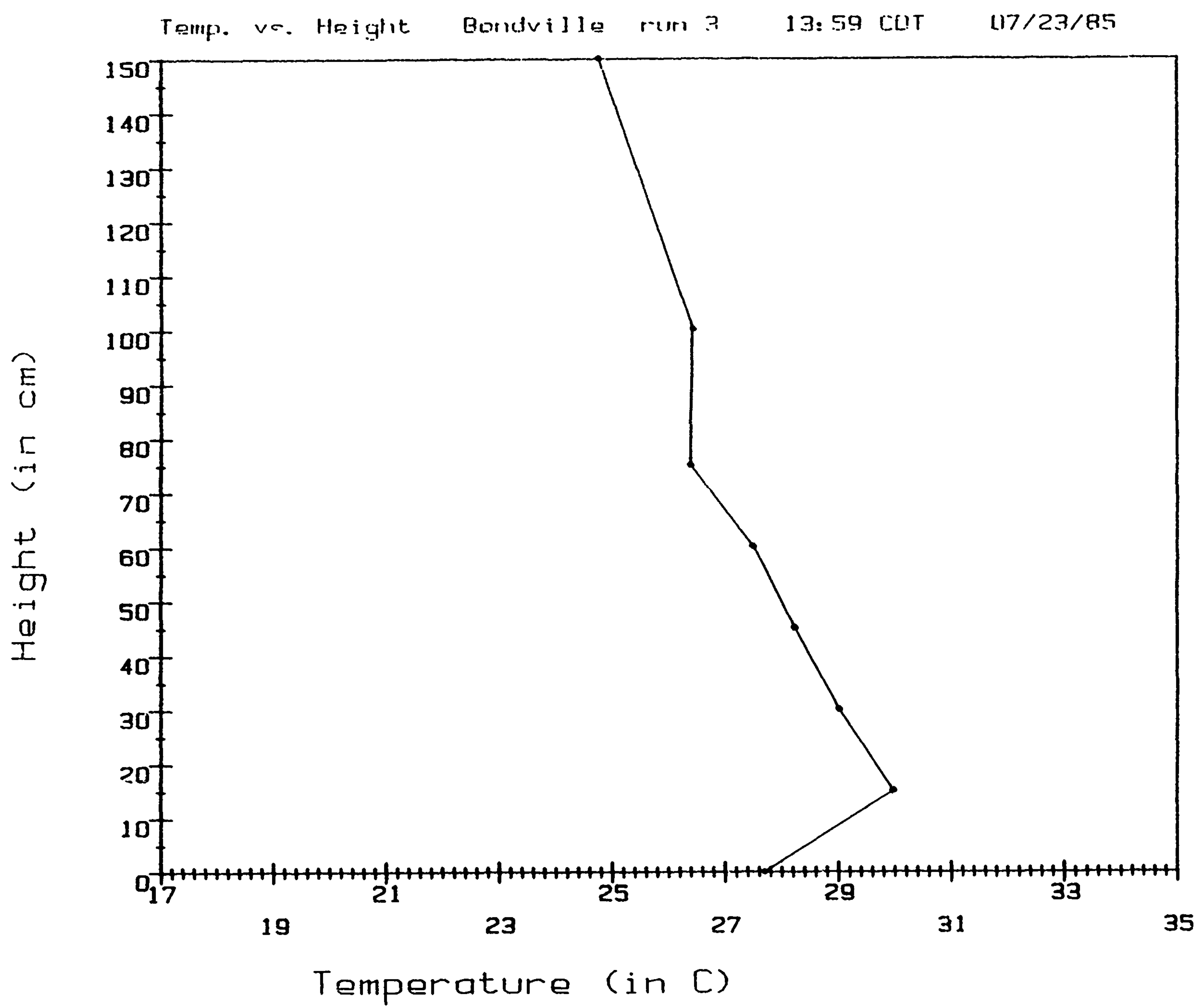


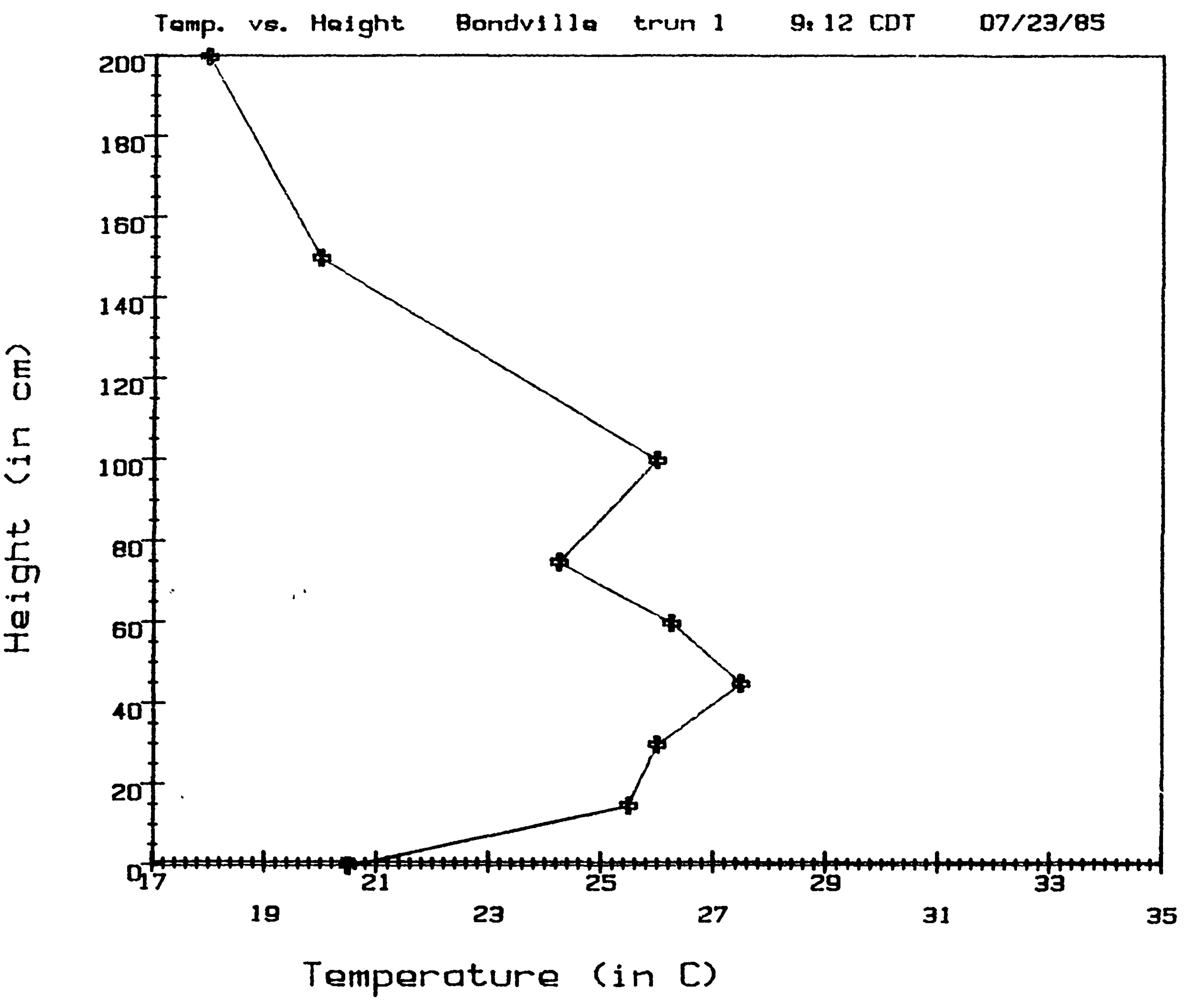
Sound Speed Profile

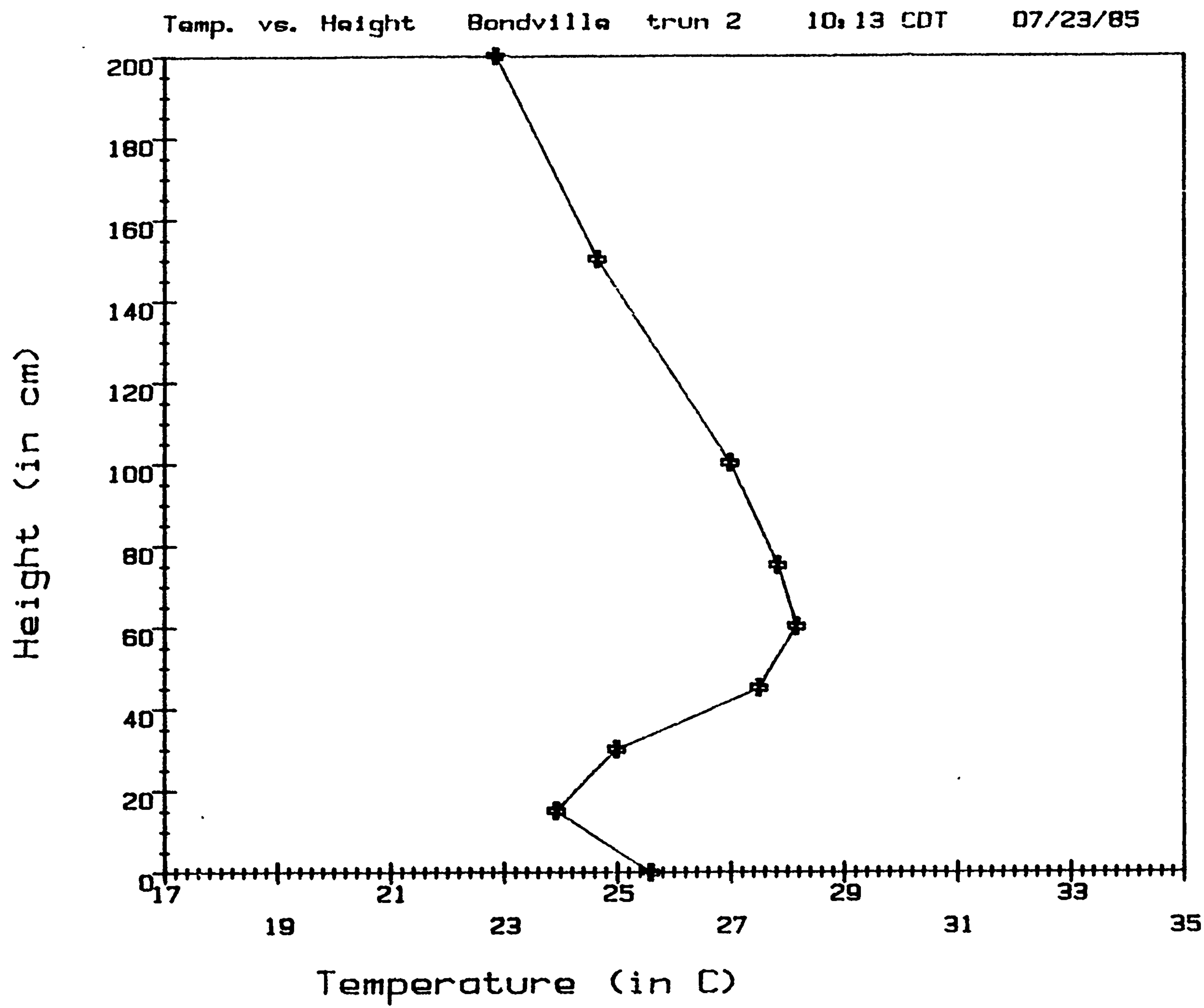
December 13, 1984
Run 4.1
Bondville, IL

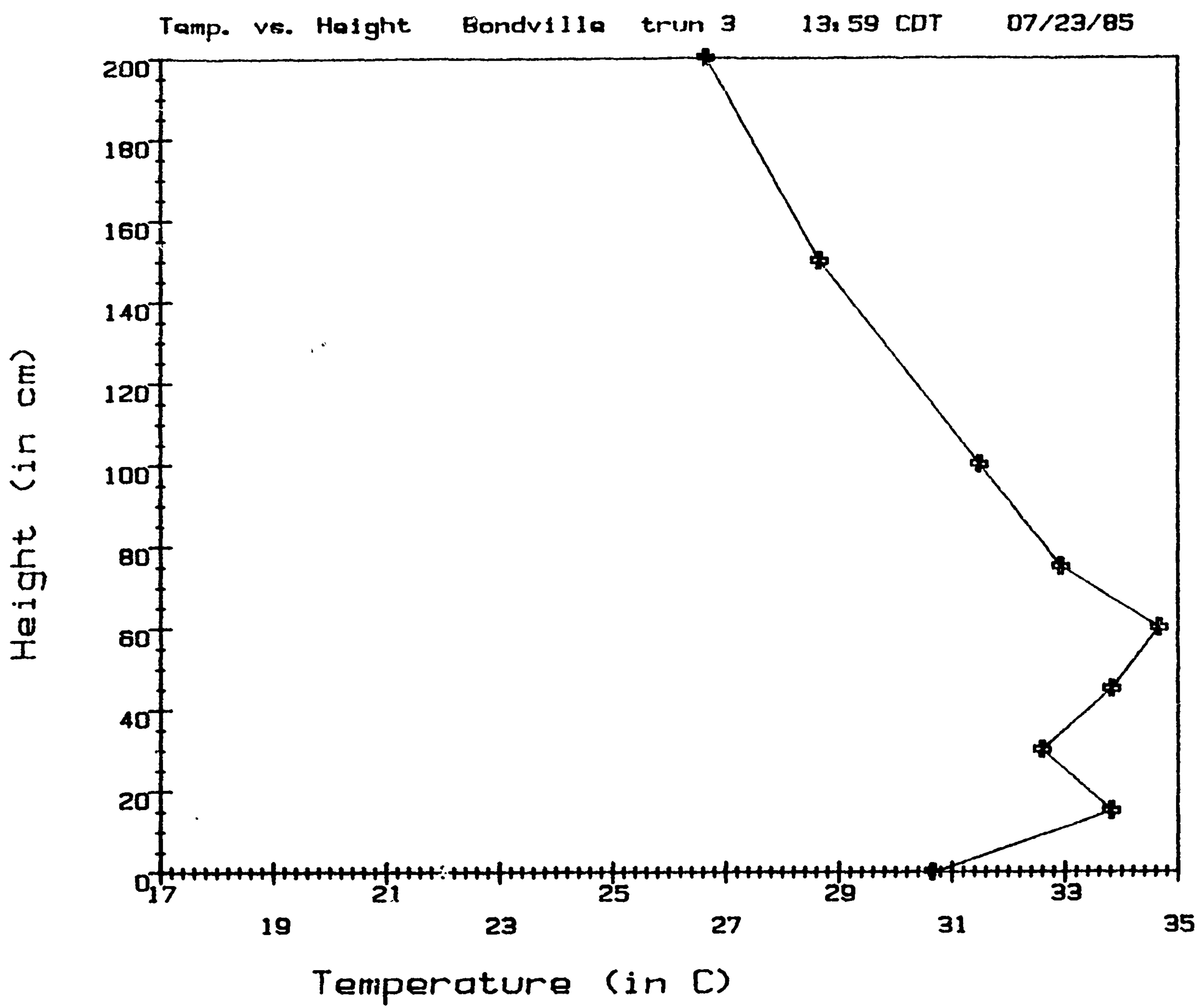

Temp. vs. Height Bondville run 1 9:12 CDT 07/23/85
Height (in cm)
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
Temperature (in C)
17
19
21
23
25
27
29
31
33
35


Temp. vs. Height Bondville run 2 10:13 CDT 07/23/85
Height (in cm)
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
17
19
21
23
25
27
29
31
33
35
Temperature (in C)

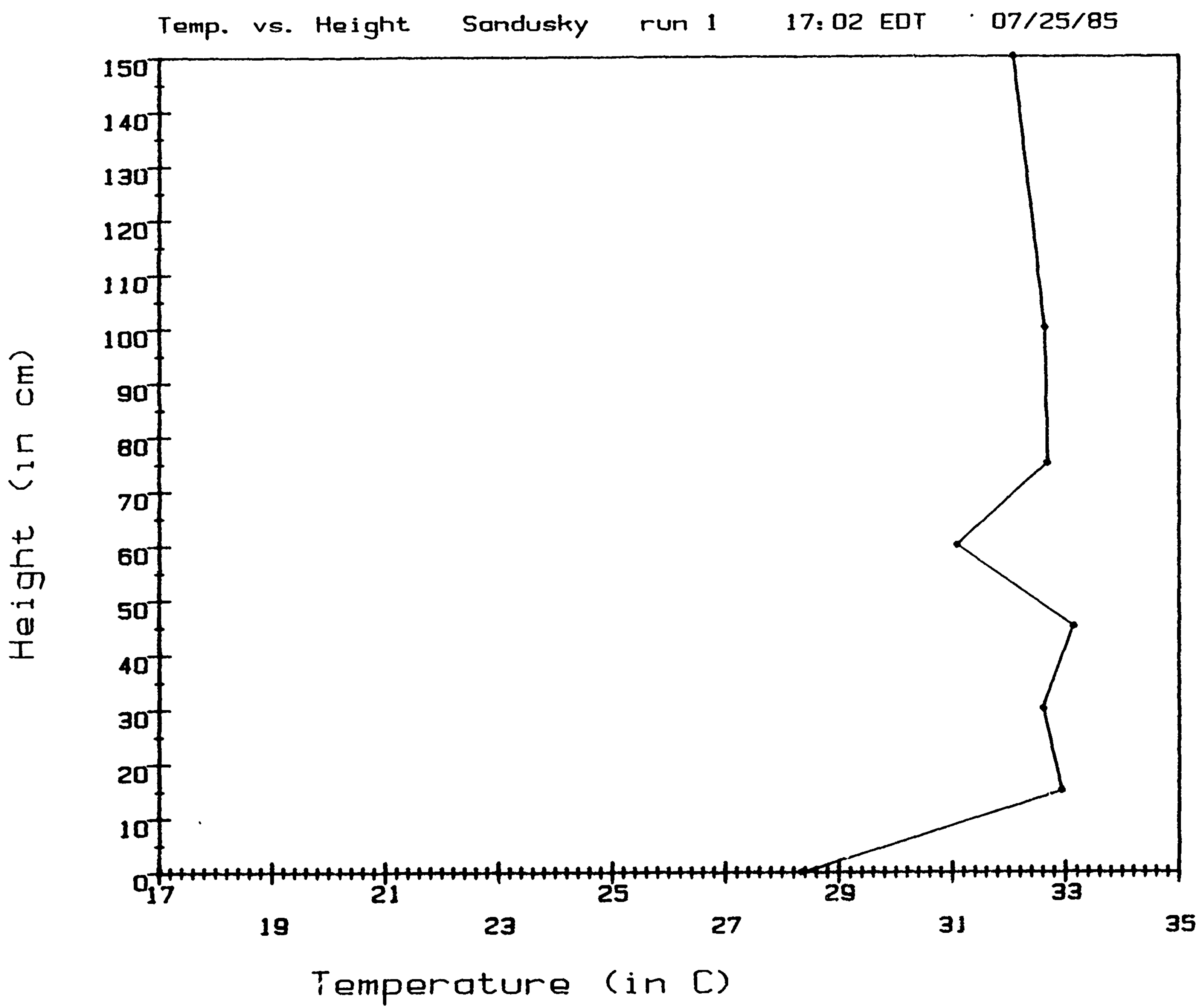

Temp. vs. Height Bondville run 3 13:59 CDT 07/23/85
Height (in cm)
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
17
19
21
23
25
27
29
31
33
35
Temperature (in C)


Temp. vs. Height Bondville trun 1 9:12 CDT 07/23/85
Height (in cm)
0
20
40
60
80
100
120
140
160
180
200
17
19
21
23
25
27
29
31
33
35
Temperature (in C)

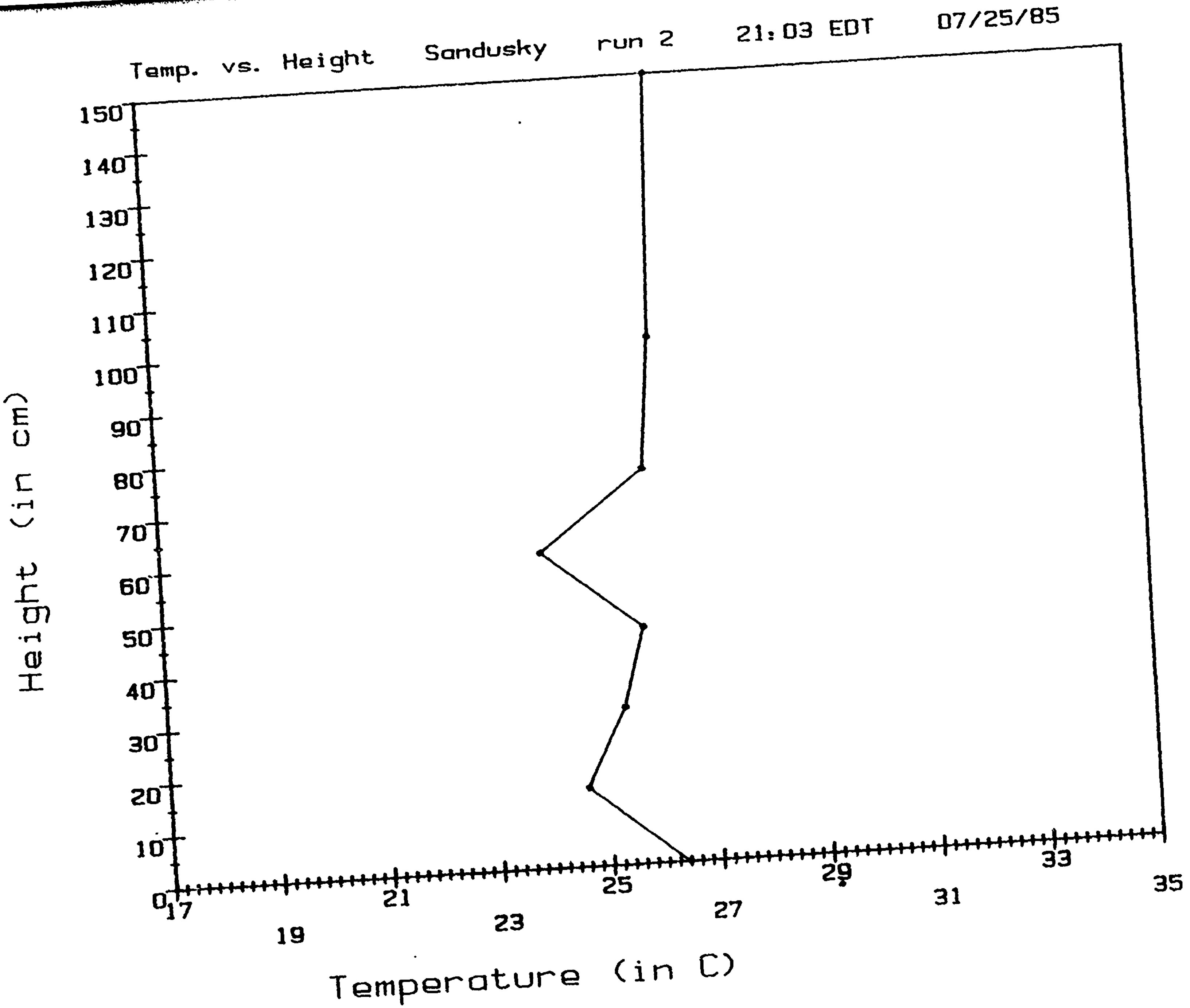

Temp. vs. Height Bondville trun 2 10:13 CDT 07/23/85
Height (in cm)
0
20
40
60
80
100
120
140
160
180
200
17
19
21
23
25
27
29
31
33
35
Temperature (in C)

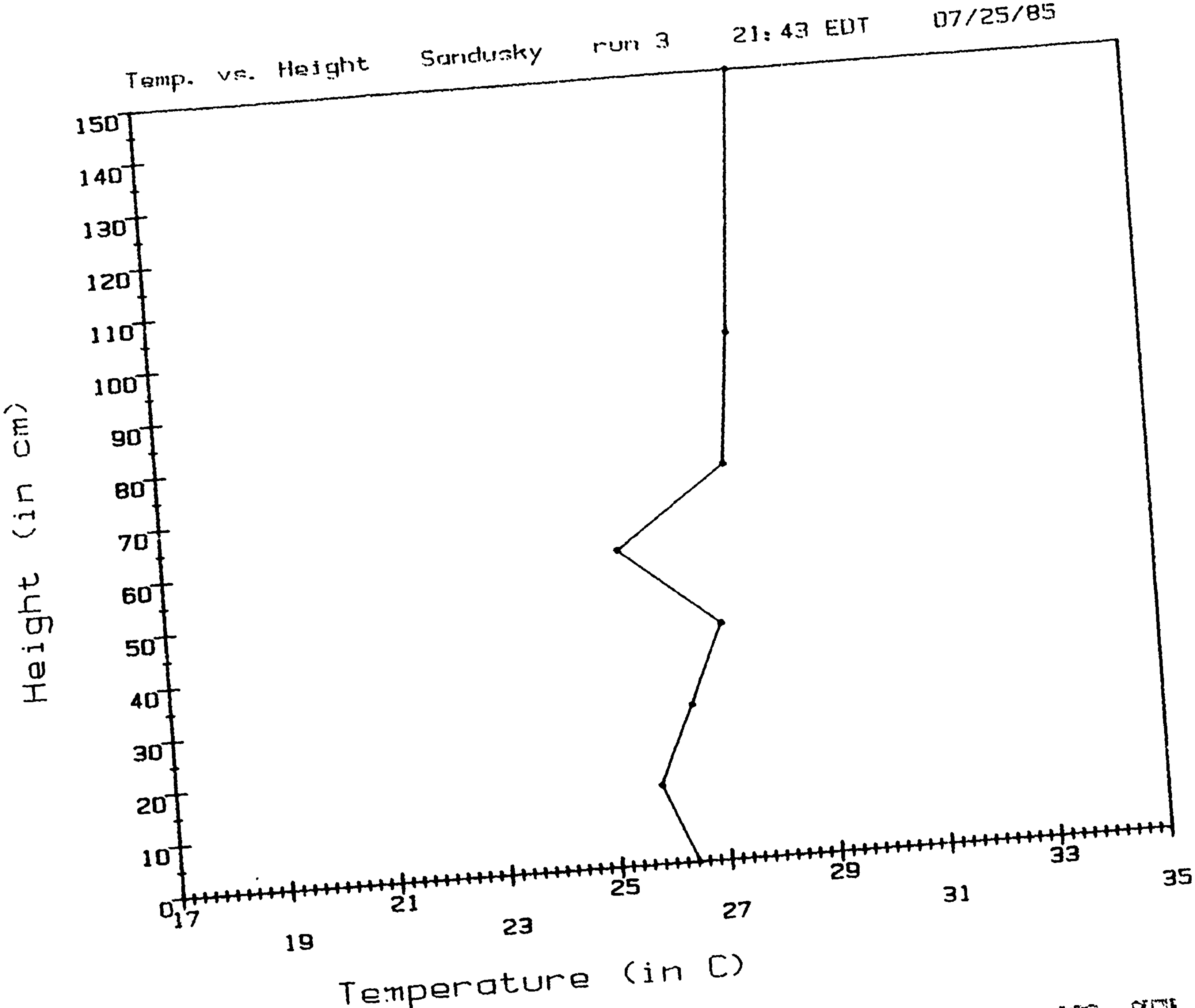


Temp. vs. Height Bondville trun 3 13:59 CDT 07/23/85
Height (in cm)
200
180
160
140
120
100
80
60
40
20
0
17
19
21
23
25
27
29
31
33
35
Temperature (in C)

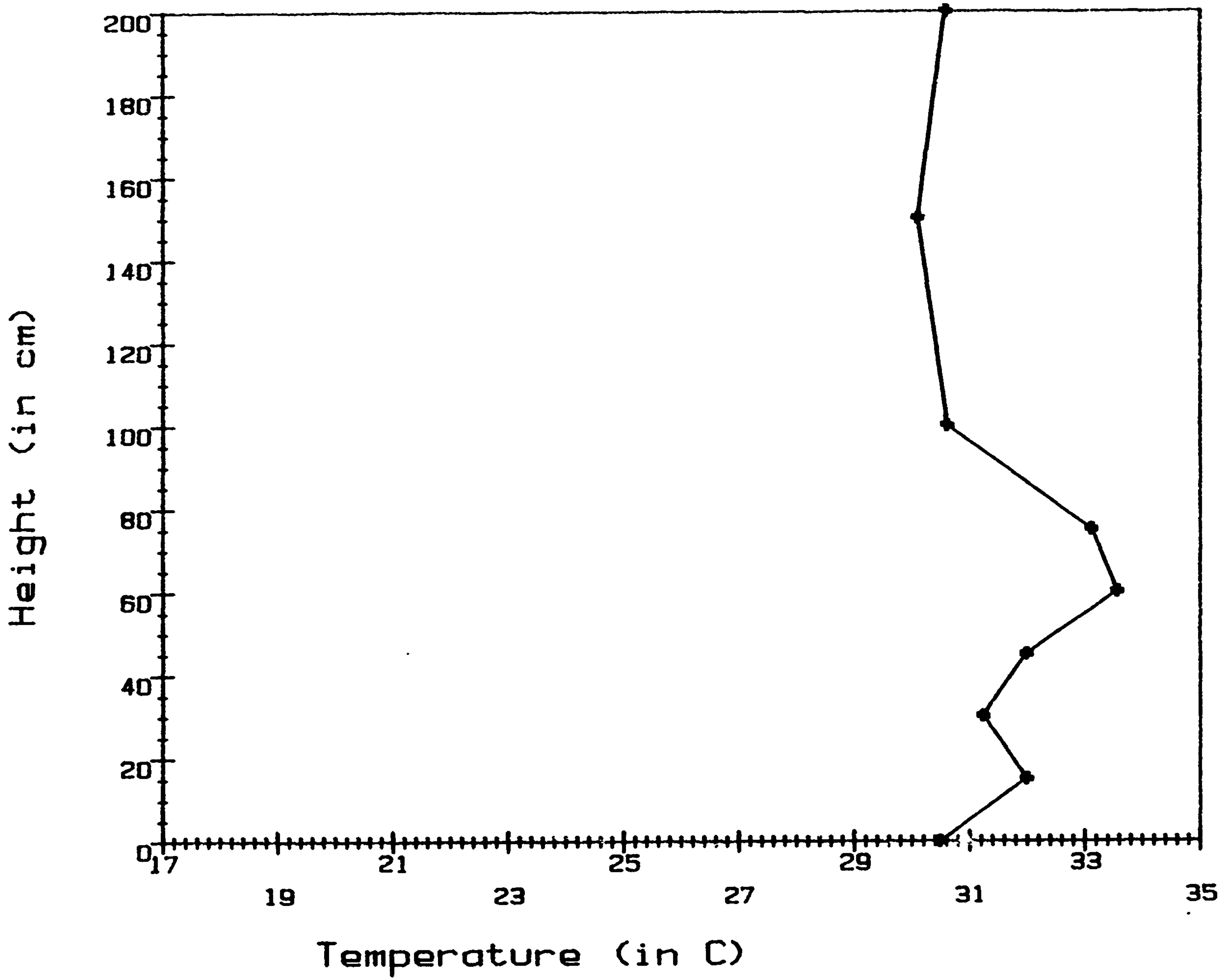


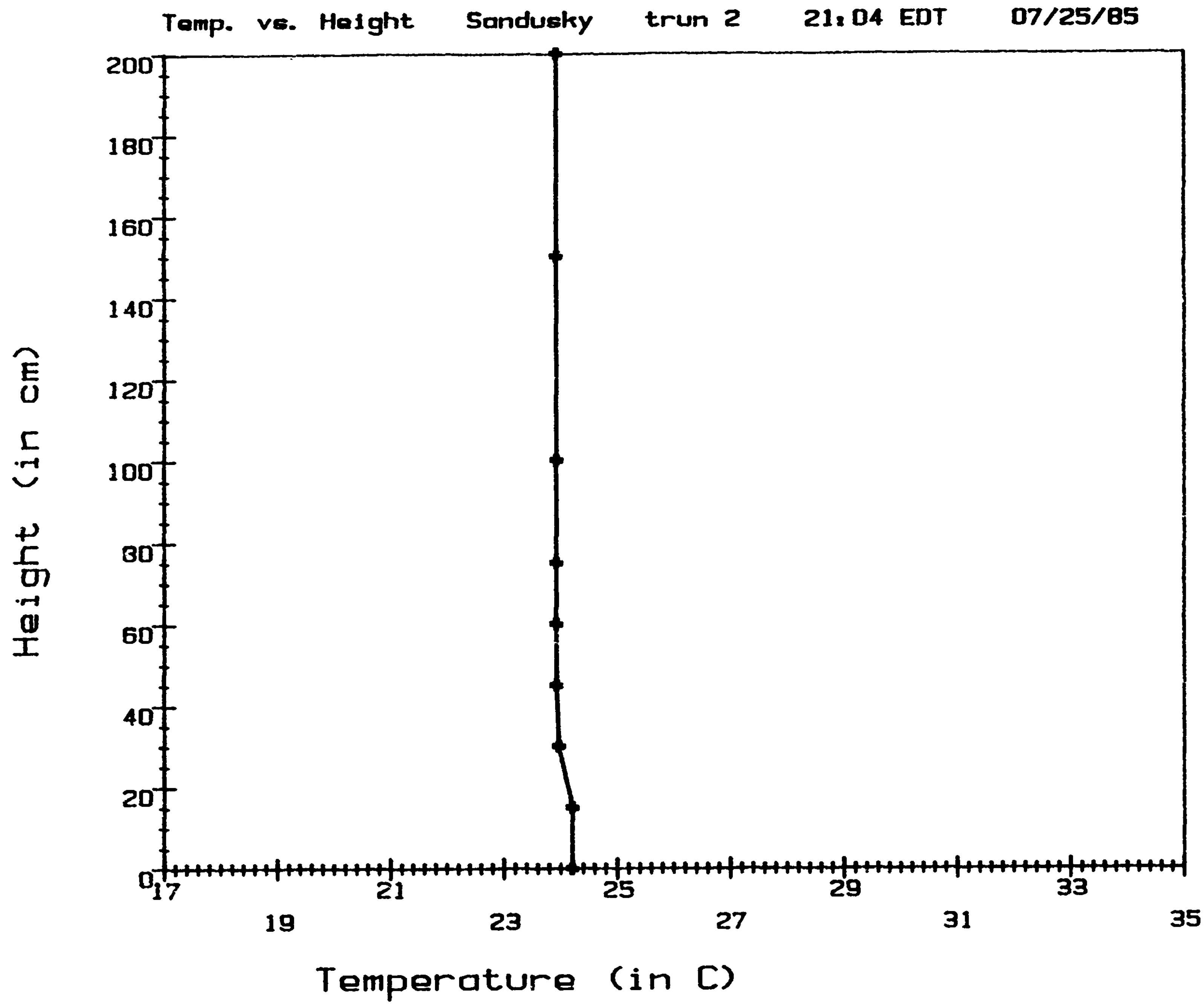
Temp. vs. Height Sandusky run 1 17:02 EDT 07/25/85
Height (in cm)
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
17
19
21
23
25
27
29
31
33
35
Temperature (in C)


Temp. vs. Height Sandusky run 2 21:03 EDT 07/25/85
Height (in cm)
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
Temperature (in C)
17
19
21
23
25
27
29
31
33
35


Temp. vs. Height   Sandusky   run 3   21:43 EDT   07/25/85
Height (in cm)
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
17
19
21
23
25
27
29
31
33
35
Temperature (in C)

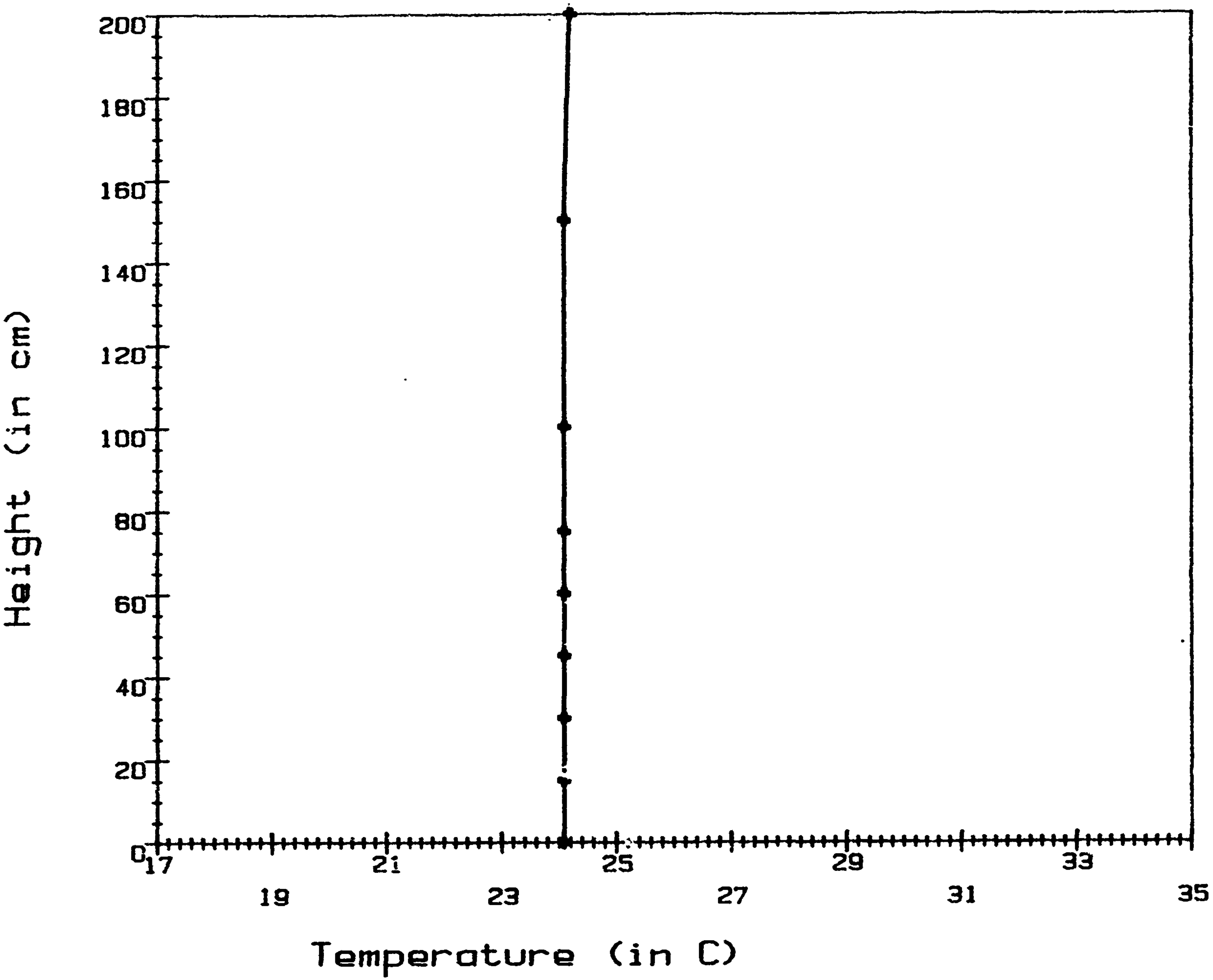
Temp. vs. Height Sandusky trun #1 17:03 EDT 07/25/85

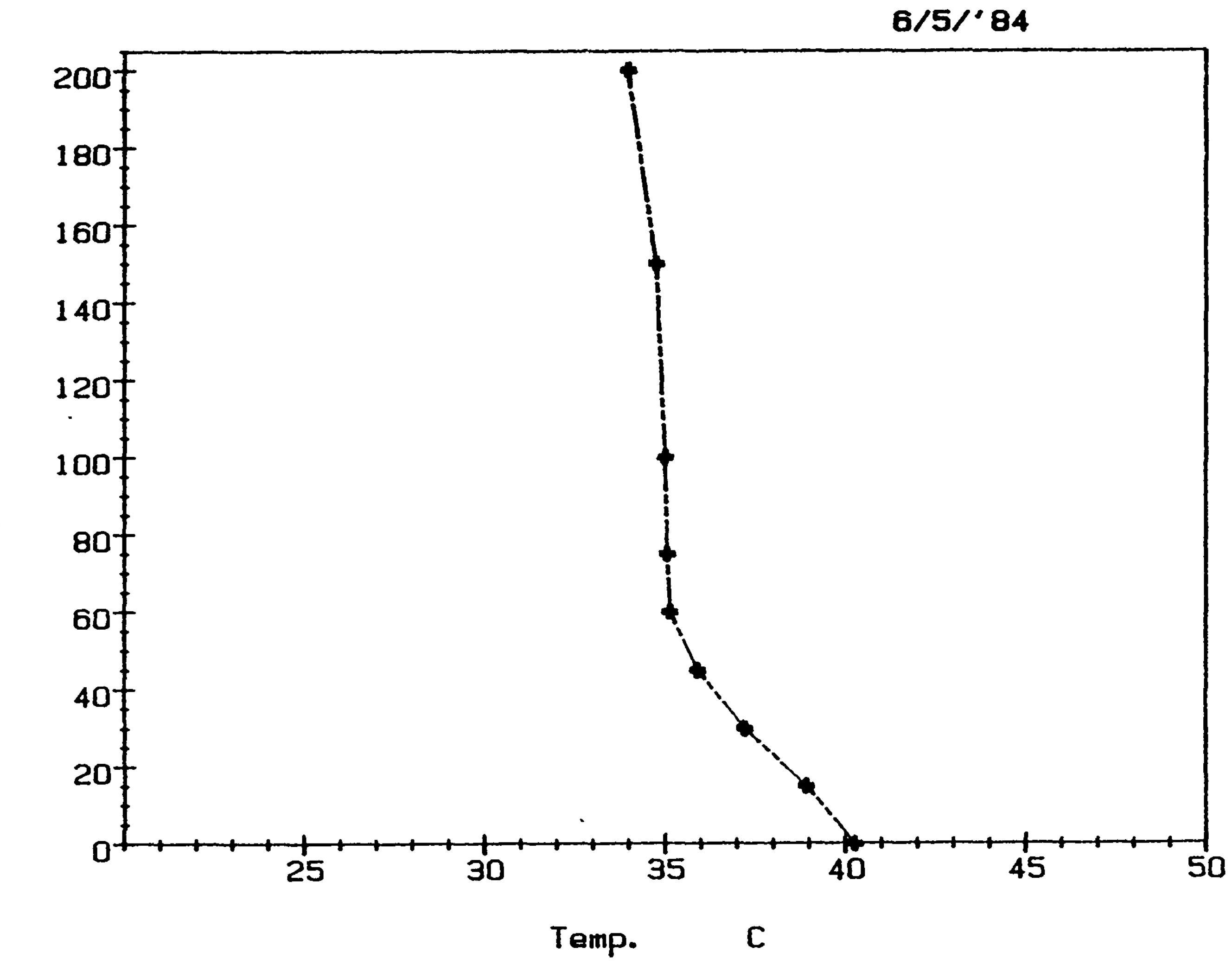




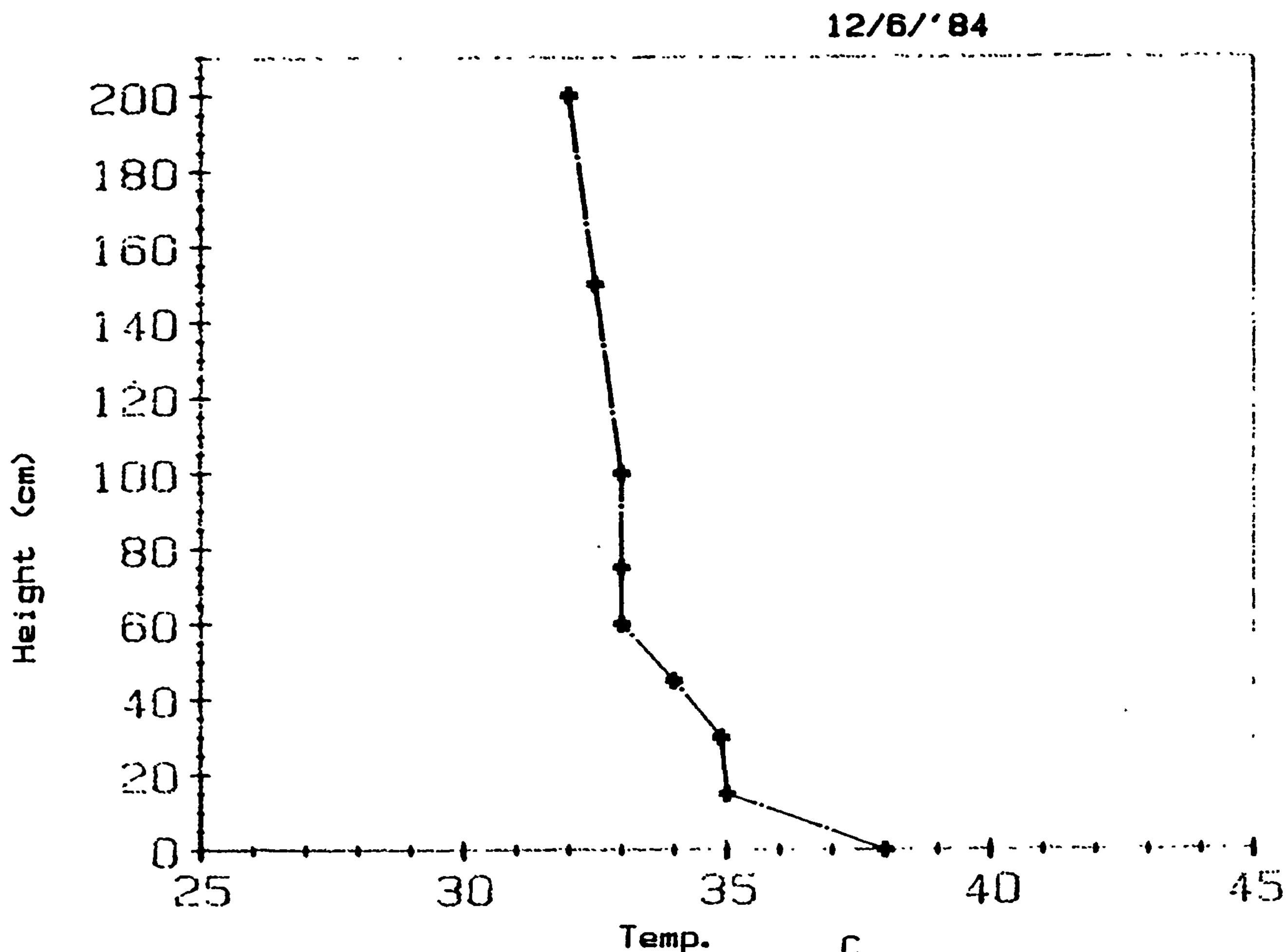
Temp. vs. Height
Sandusky
trun 2
21:04 EDT
07/25/85
Height (in cm)
200
180
160
140
120
100
80
60
40
20
0
17
19
21
23
25
27
29
31
33
35
Temperature (in C)

Temp. vs. Height Sandusky trun 3 ~~17:03~~ 21:45 EDT 07/25/85

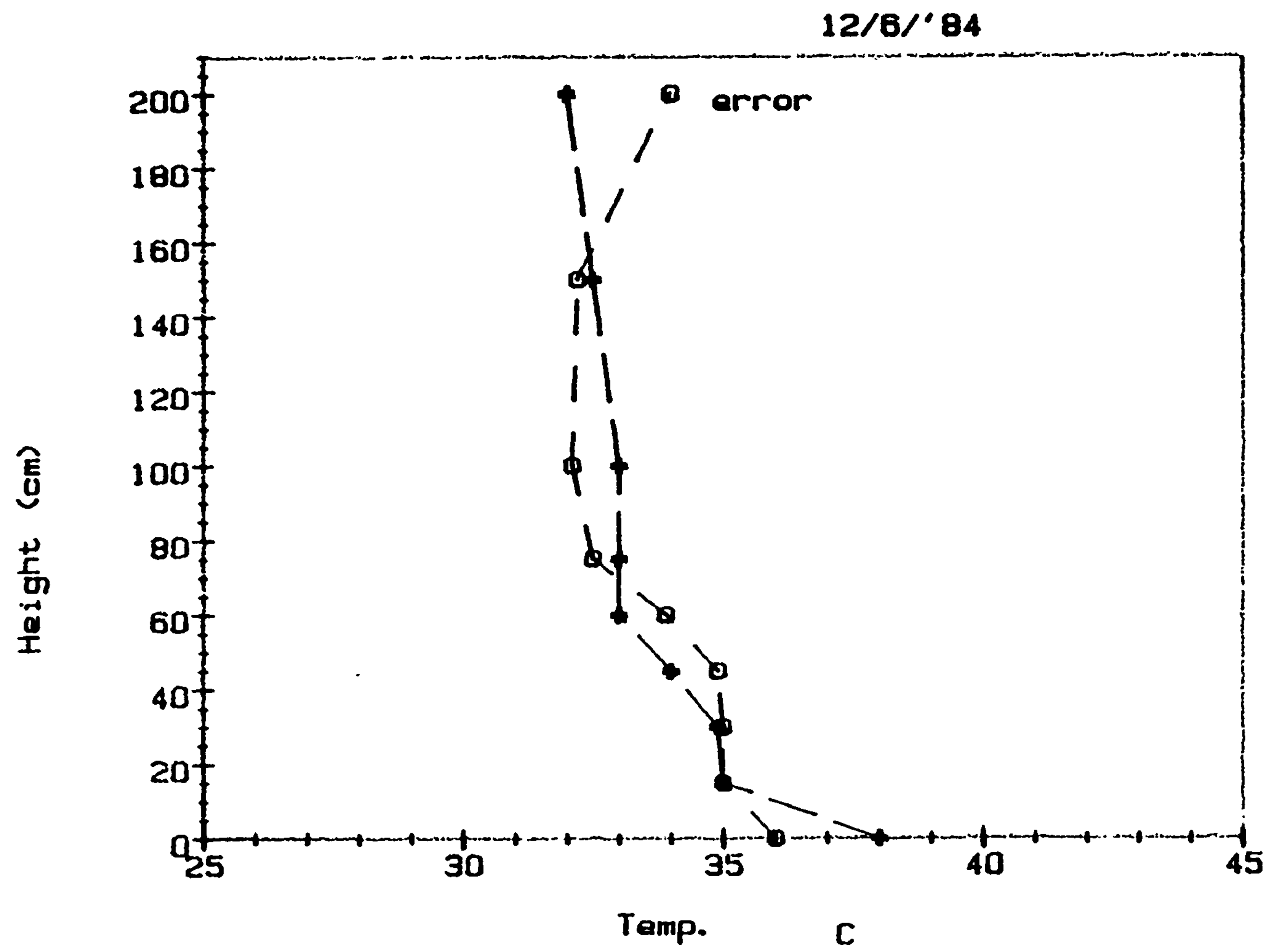


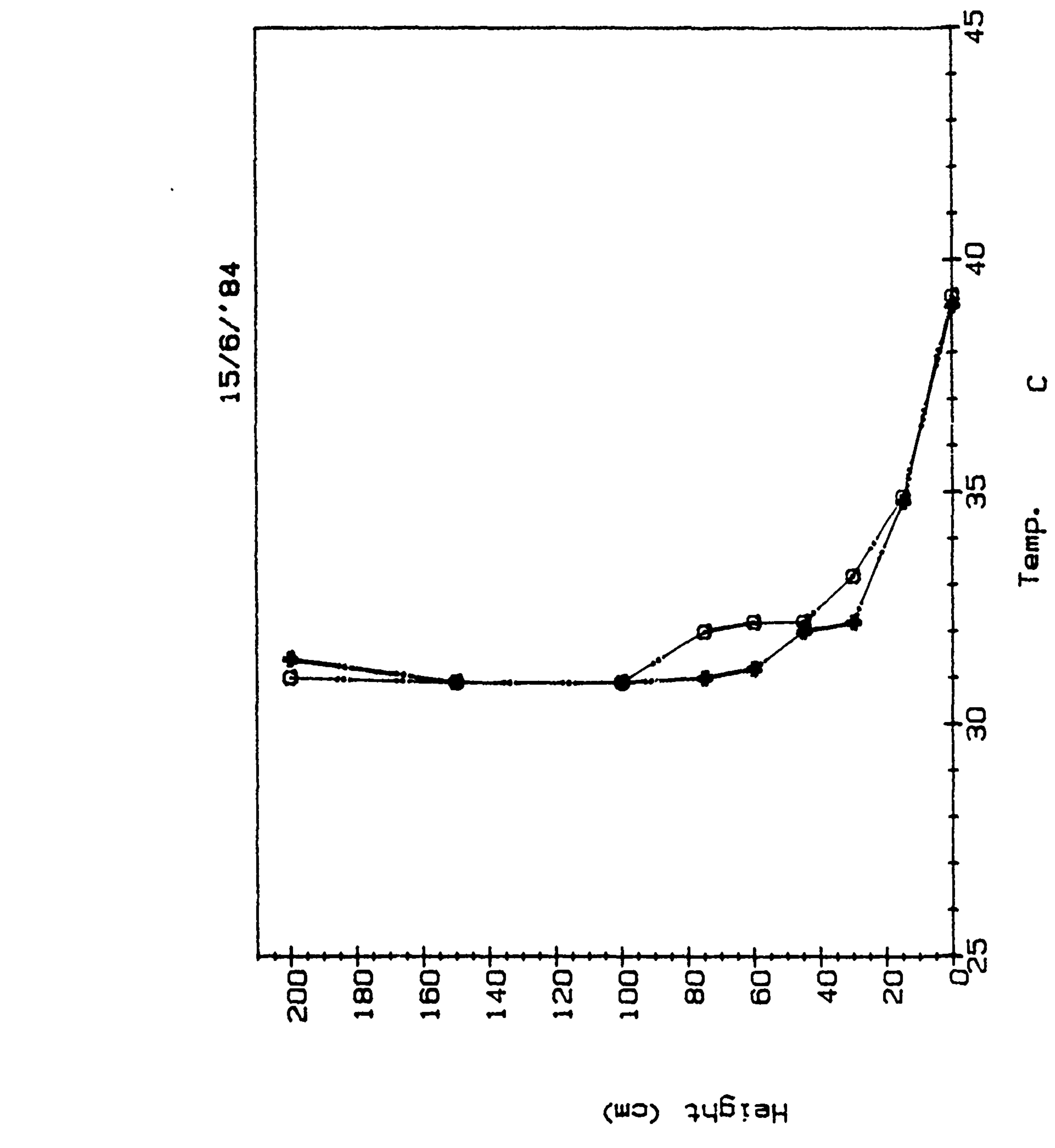

6/5/'84
Height (cm)
200
180
160
140
120
100
80
60
40
20
0
25
30
35
40
45
50
Temp. C

DATA5



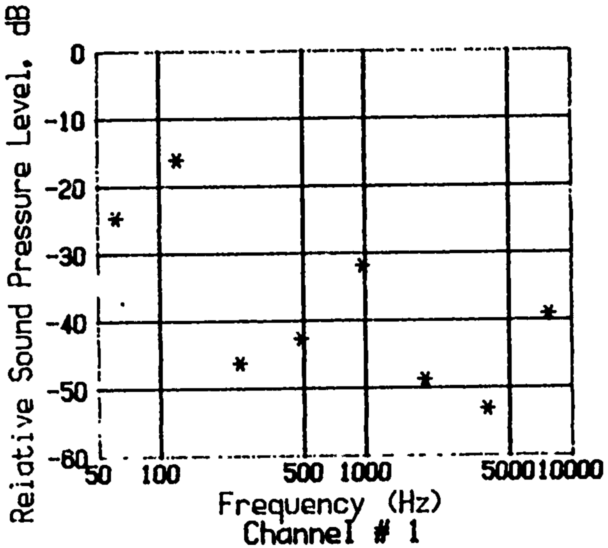
12/6/'84
Height (cm)
200
180
160
140
120
100
80
60
40
20
0
25
30
35
40
45
Temp. C

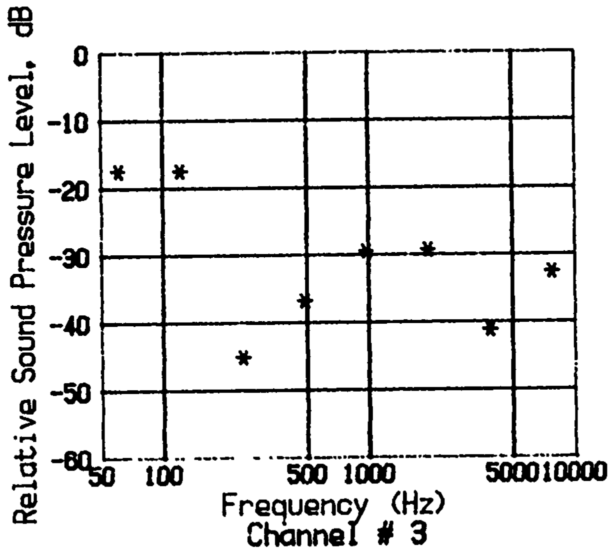

12/6/'84
error
Height (cm)
200
180
160
140
120
100
80
60
40
20
0
25
30
35
40
45
Temp. C

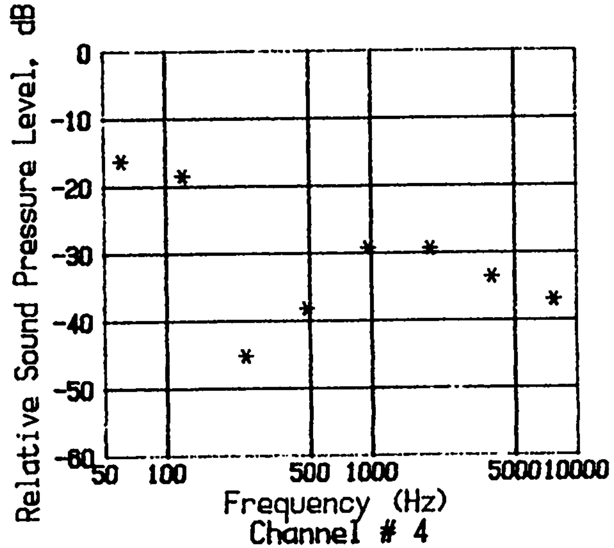

15/6/'84
Height (cm)
Temp. C
0
20
40
60
80
100
120
140
160
180
200
25
30
35
40
45

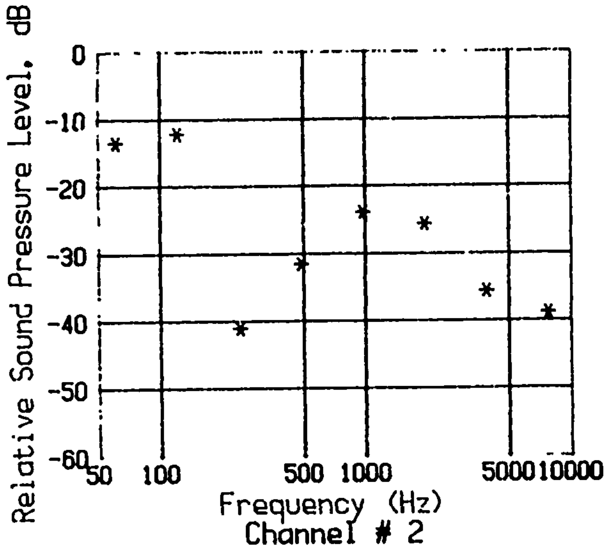
APPENDIX D

Plots of the relative sound pressure level for each microphone and each experiment.

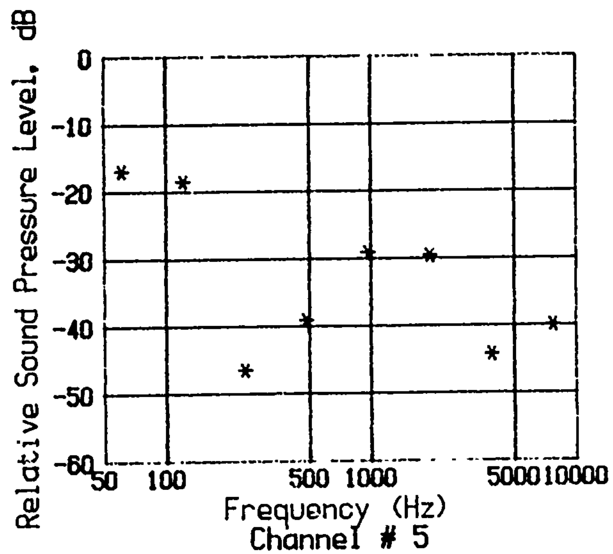


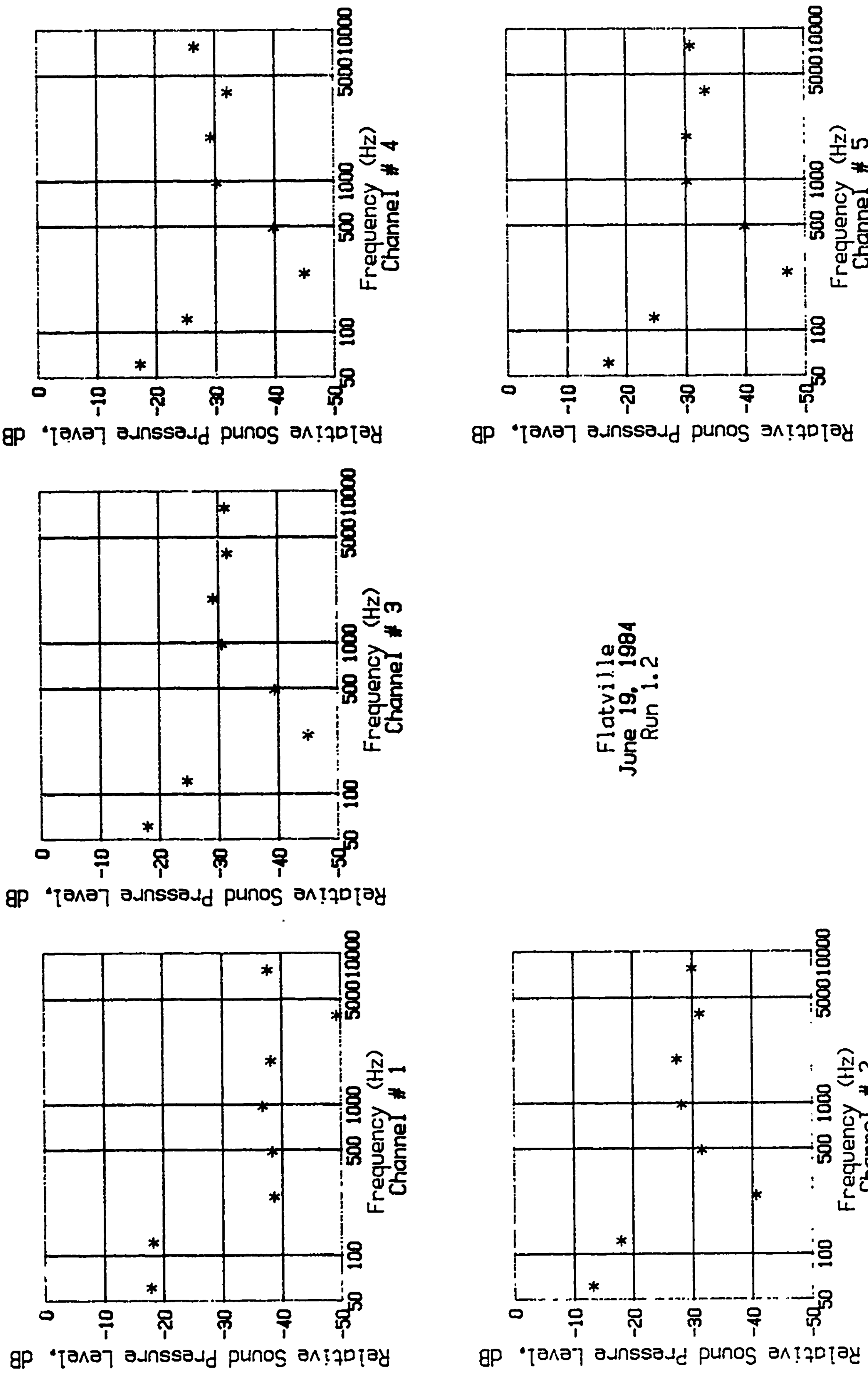


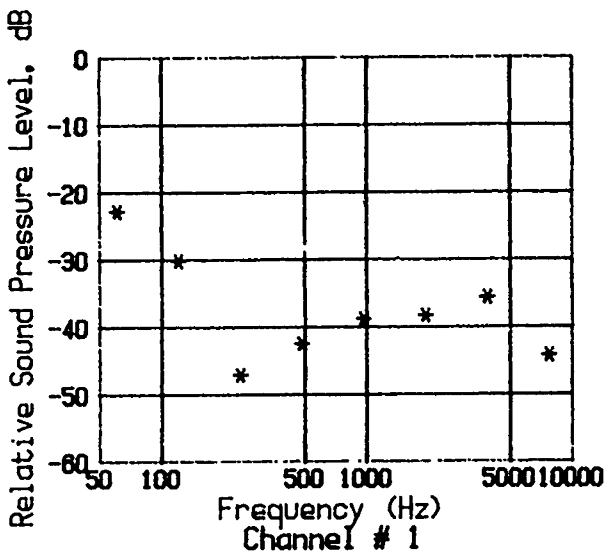


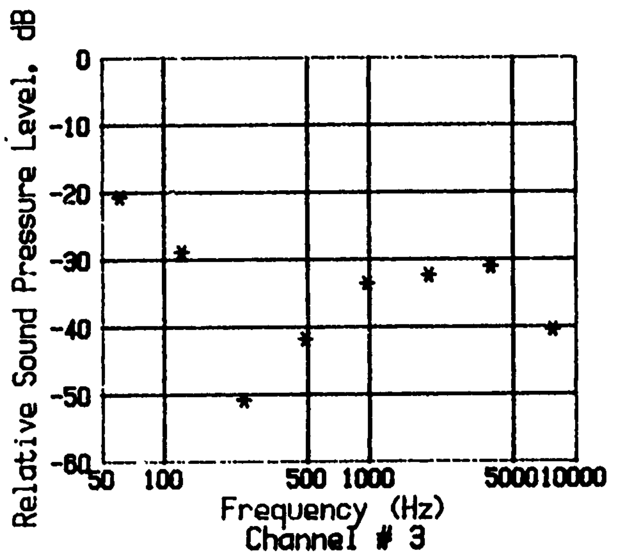


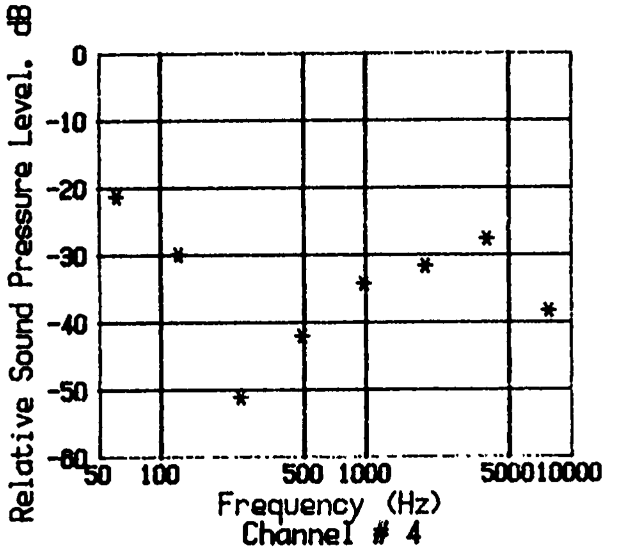
Flatville
June 19, 1984
Run 1.1

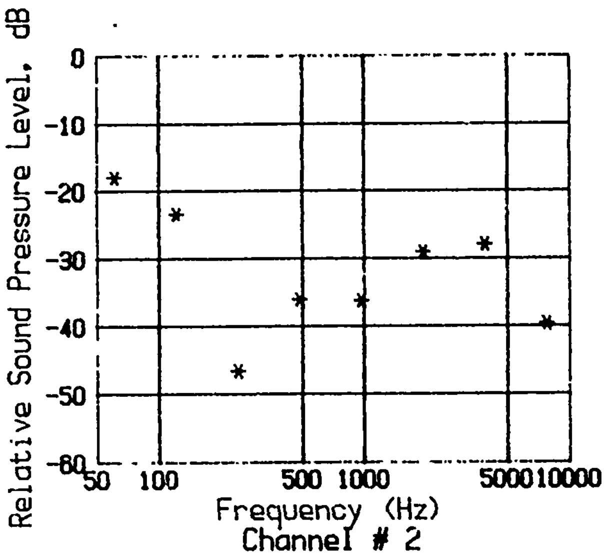



Relative Sound Pressure Level, dB
Frequency (Hz)
Channel # 1
Channel # 2
Channel # 3
Channel # 4
Channel # 5
Flatville
June 19, 1984
Run 1.2

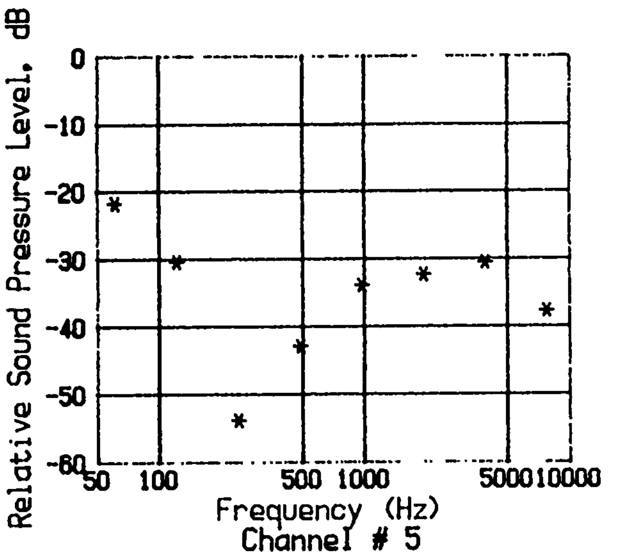


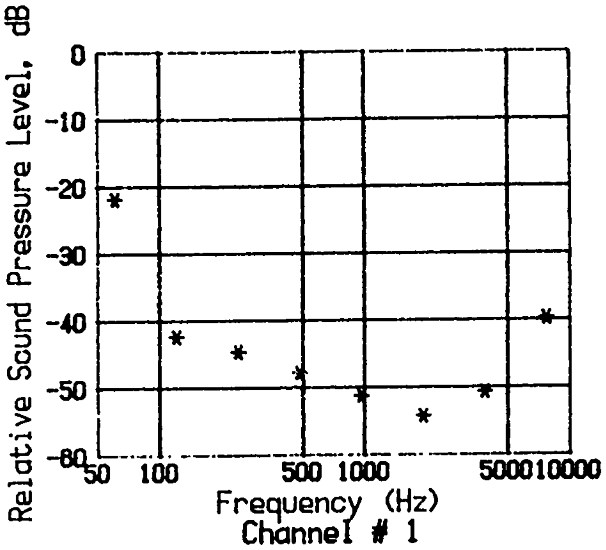


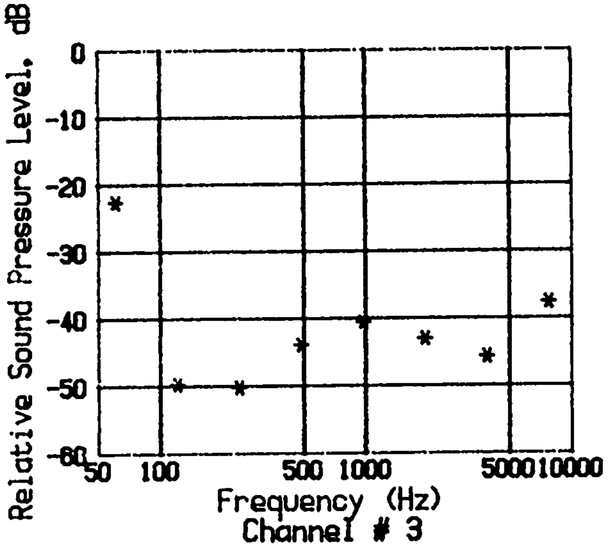


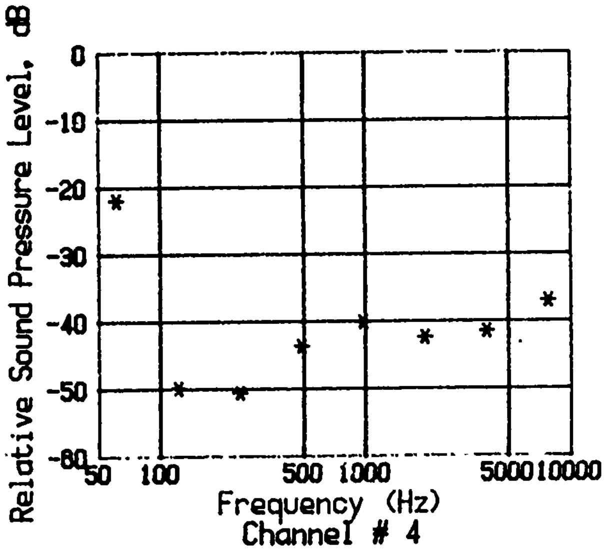


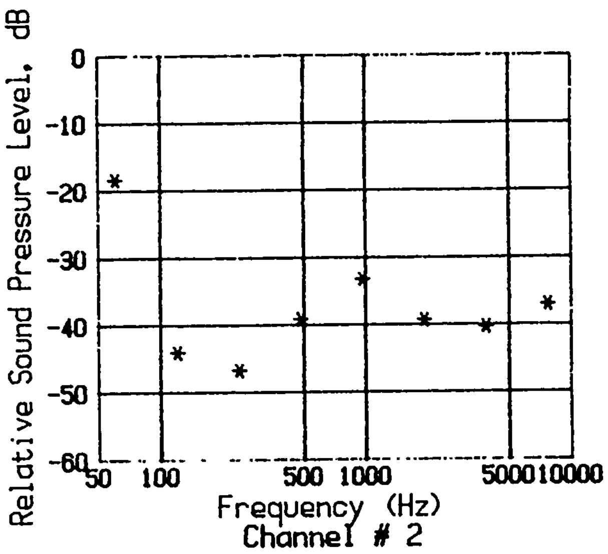
Flatville
June 19, 1984
Run 2.1



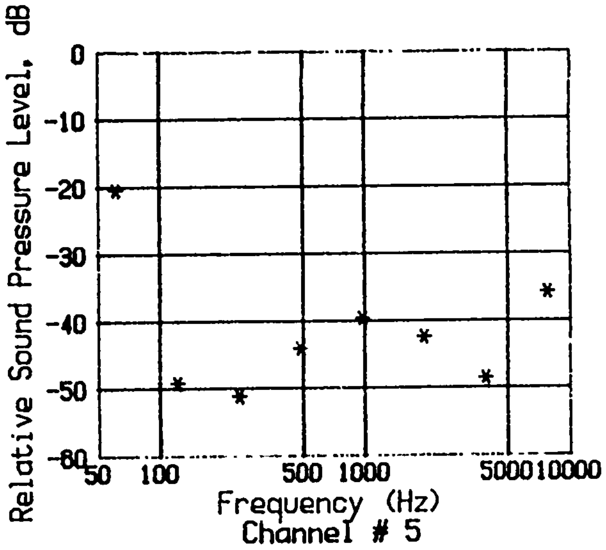


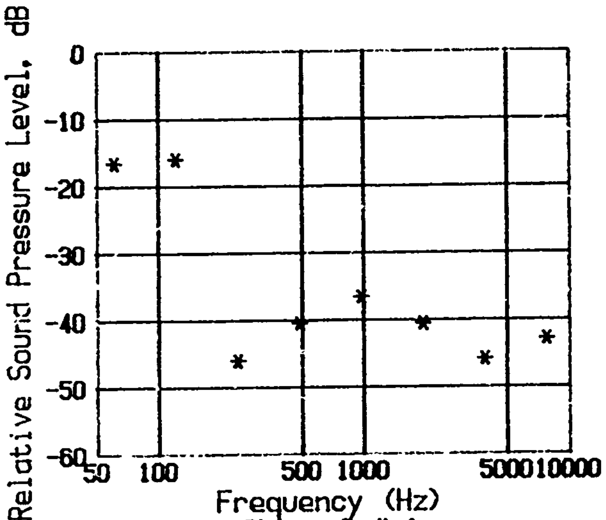




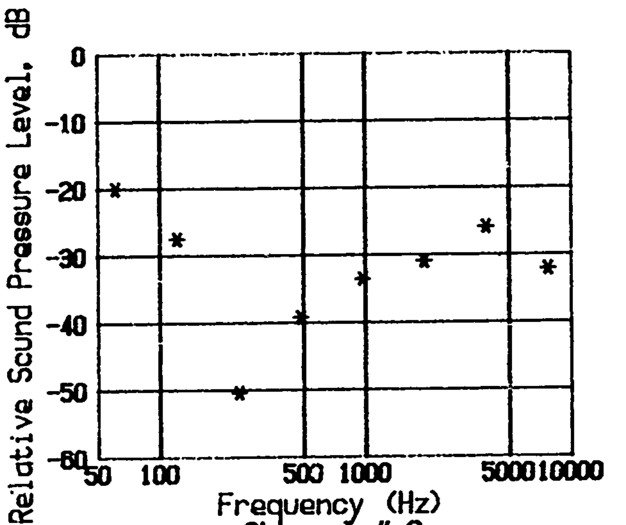


Flatville
June 19, 1984
Run 2.2

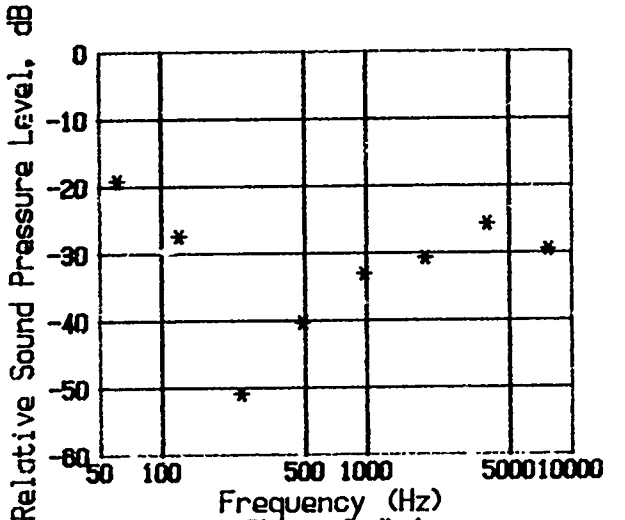




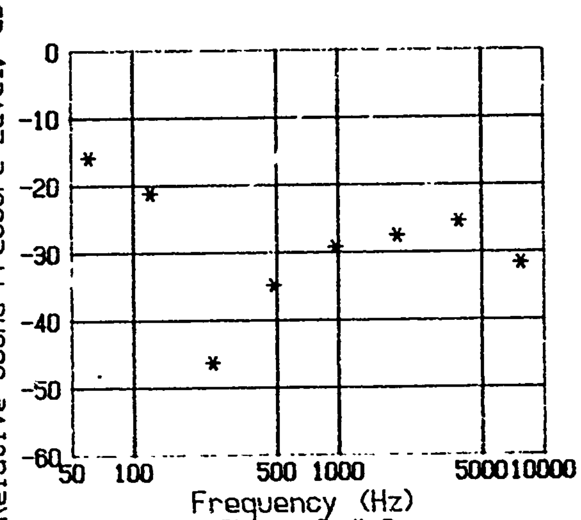
Channel # 1



Channel # 3

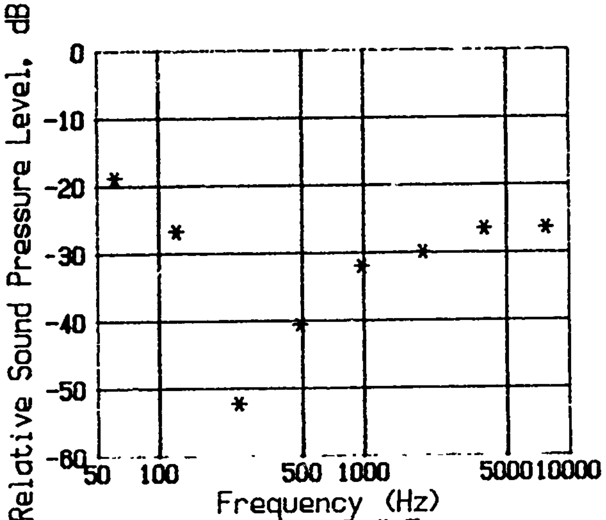


Channel # 4

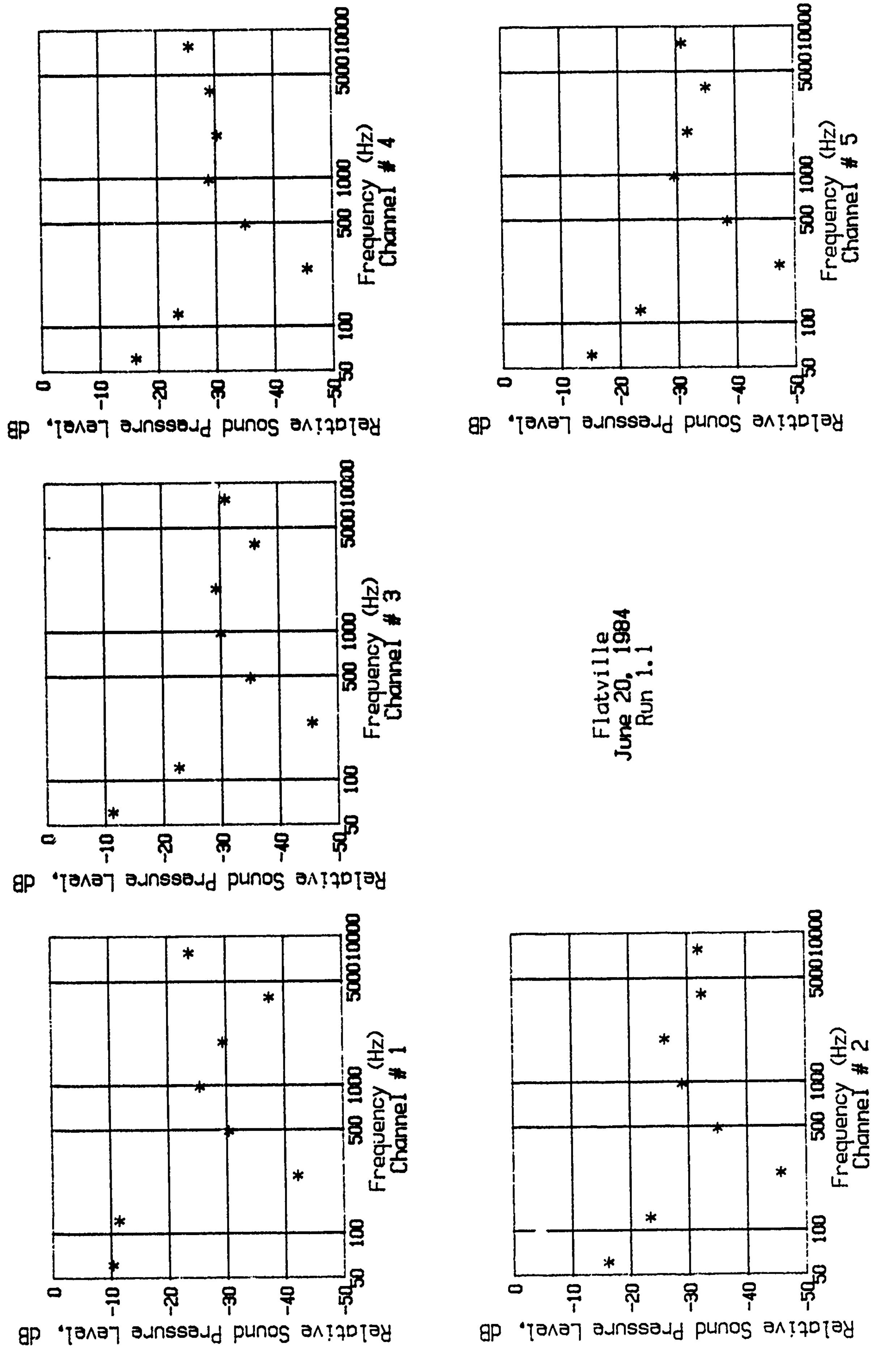


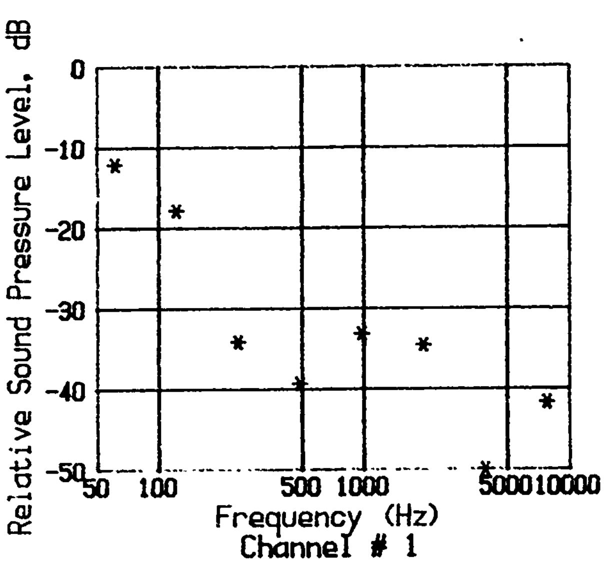
Channel # 2

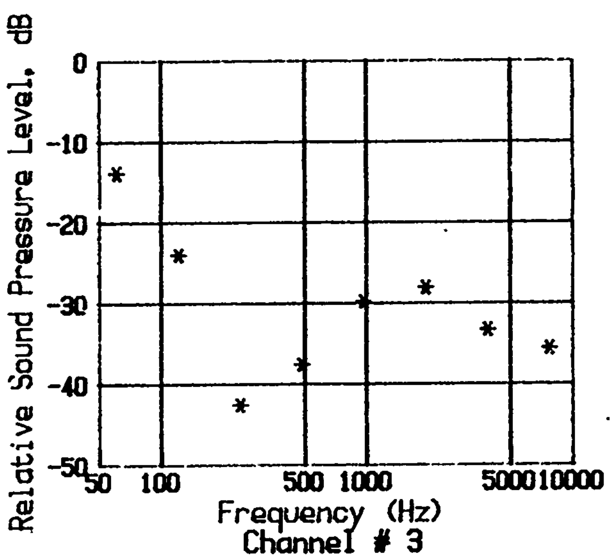
Flatville
June 19, 1984
Run 2.3

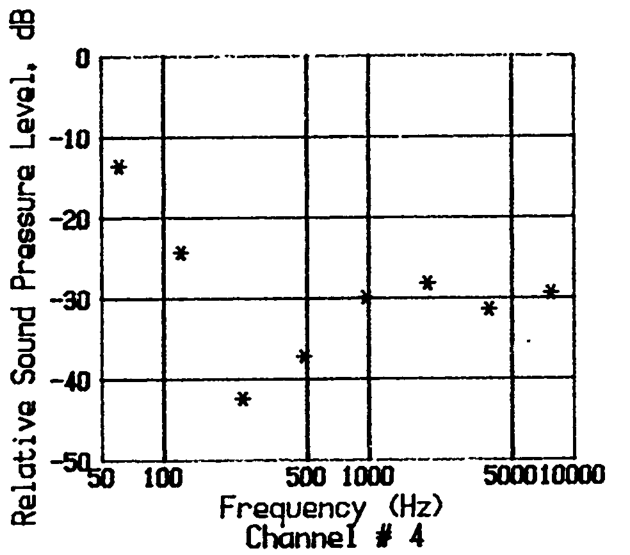


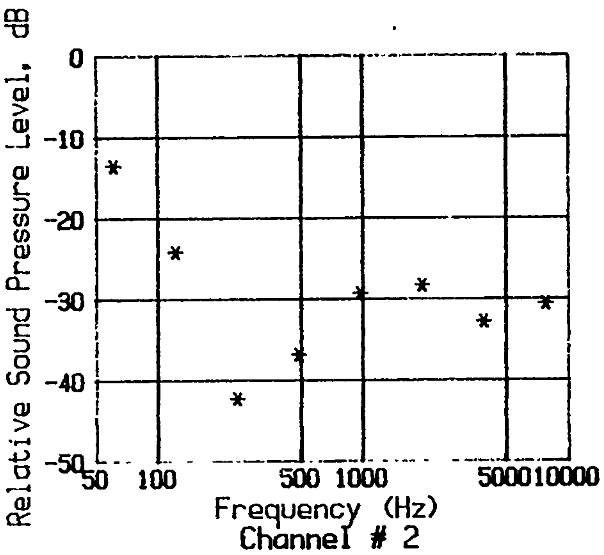
Channel # 5


Relative Sound Pressure Level, dB
Frequency (Hz)
Channel # 1
Channel # 2
Channel # 3
Channel # 4
Channel # 5
Flatville
June 20, 1984
Run 1.1

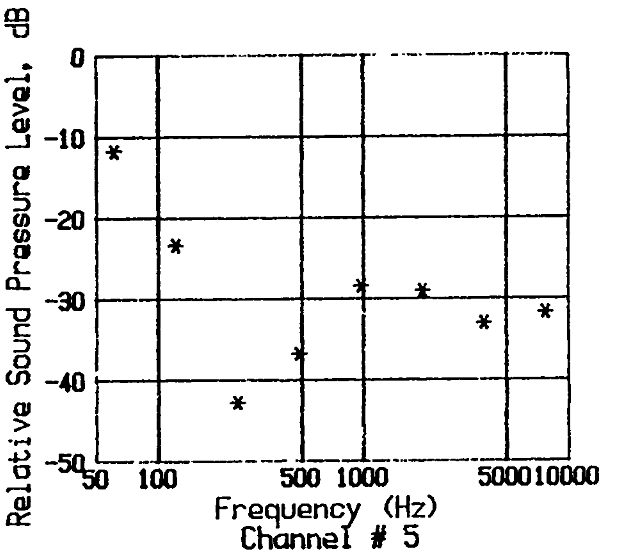


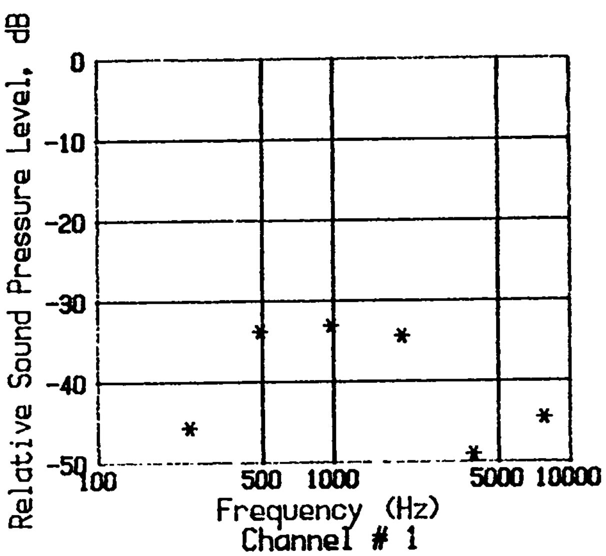


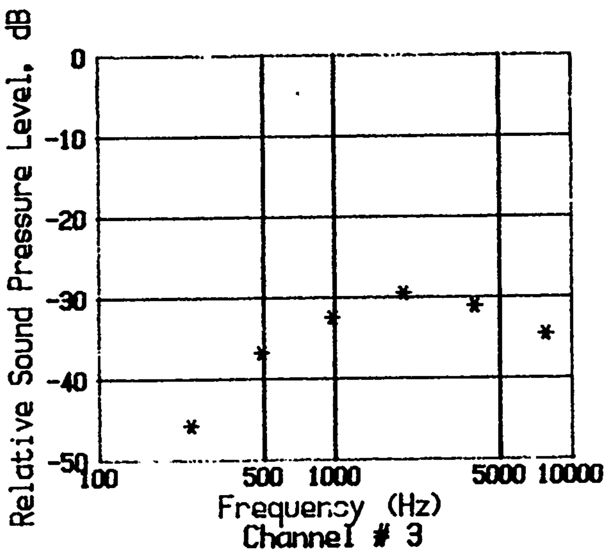


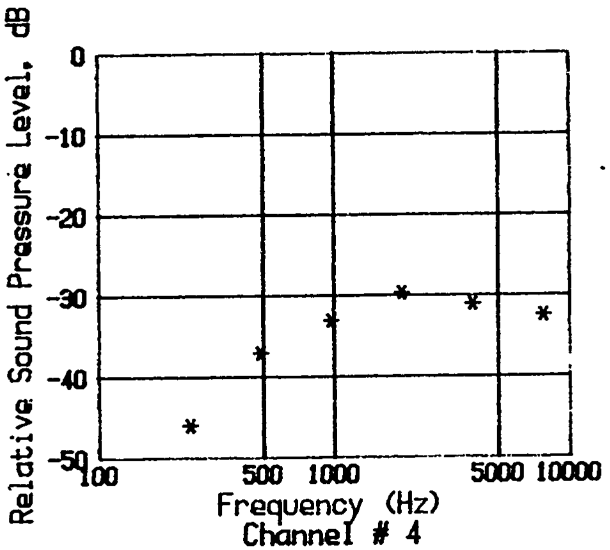


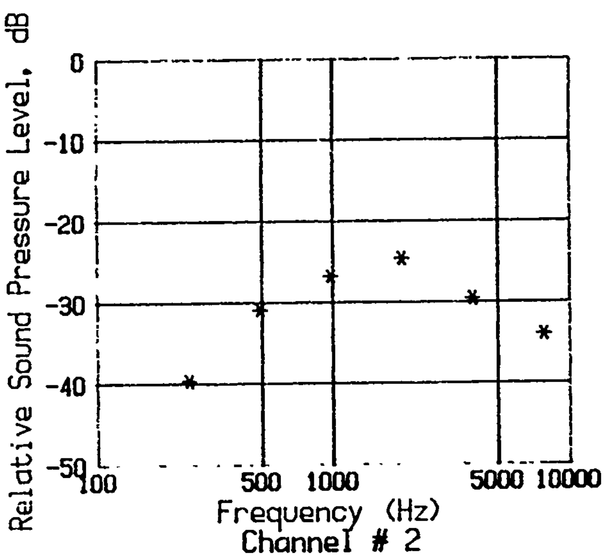
Flatville
June 20, 1984
Run 1.2



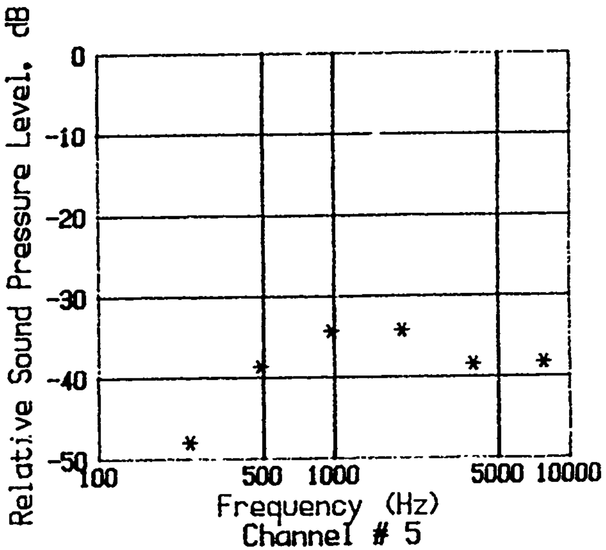


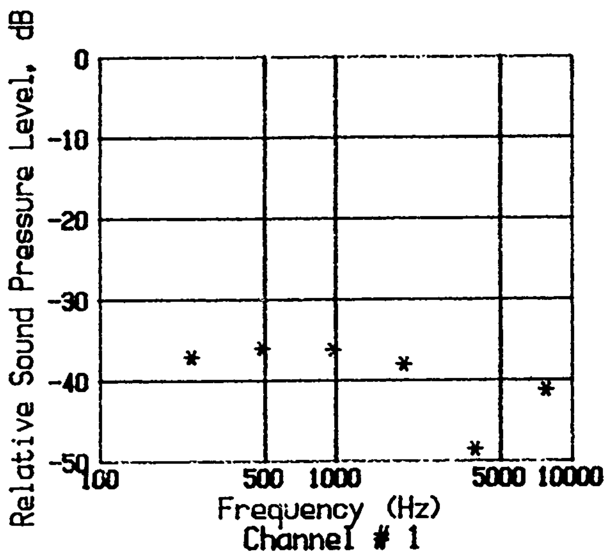


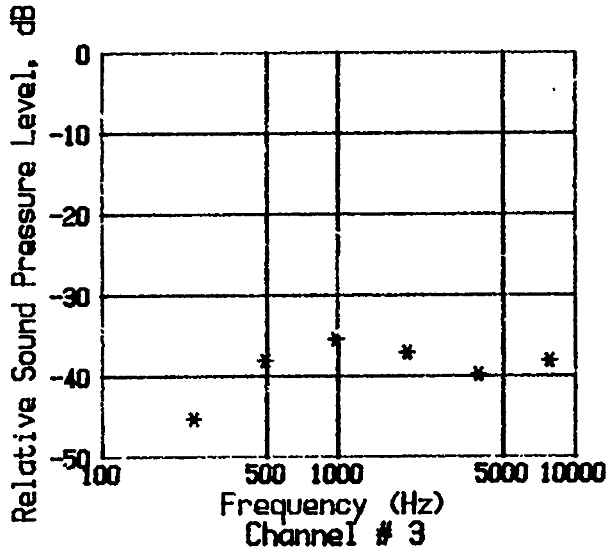


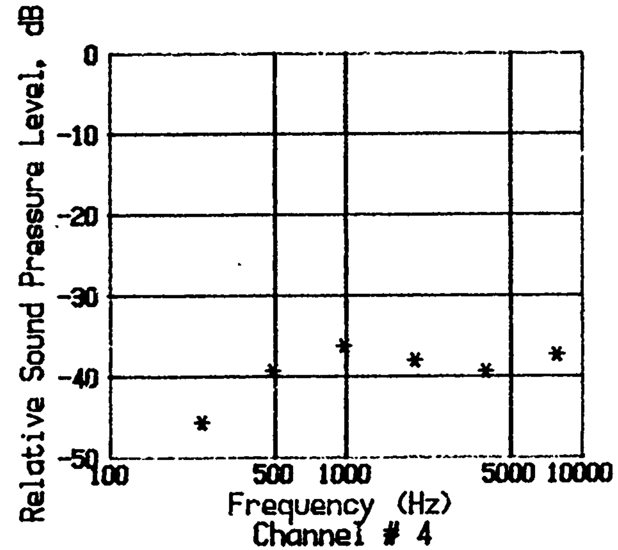


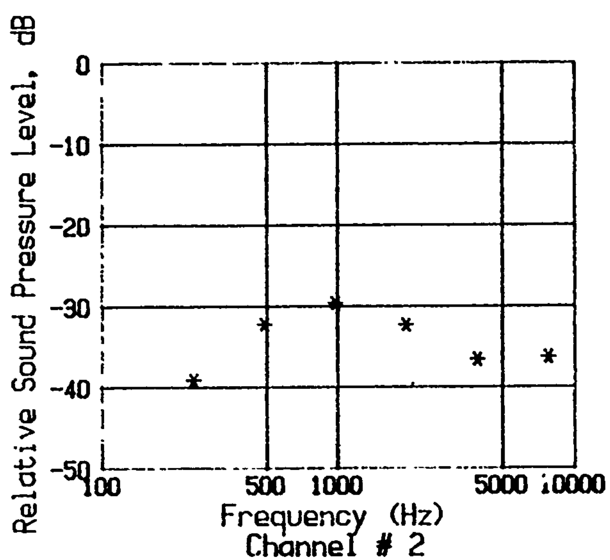
Flatville
June 20, 1984
Run 2.1



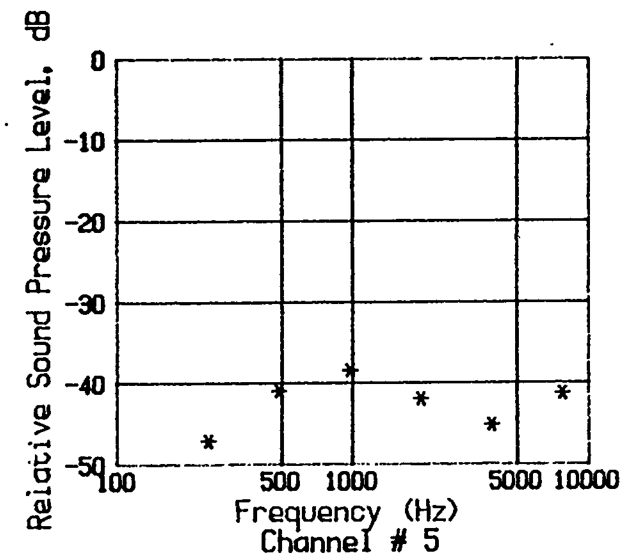


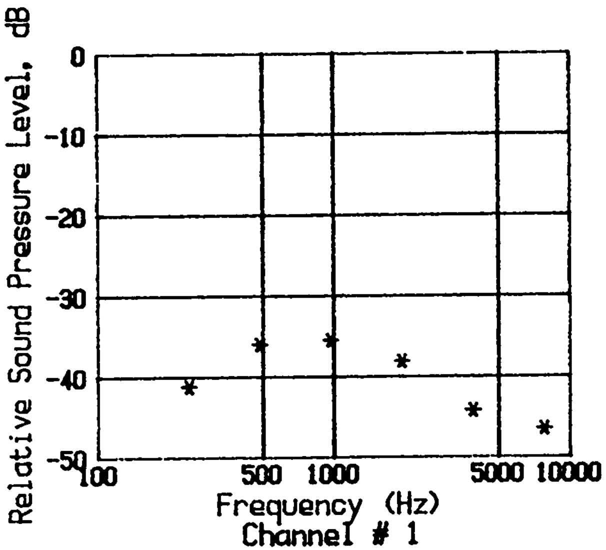


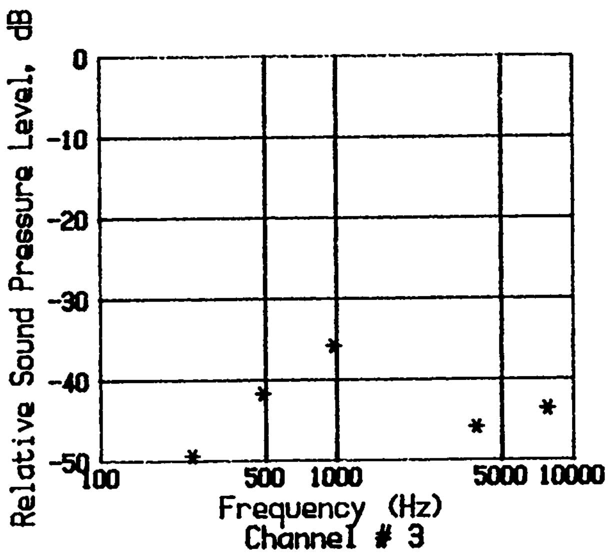


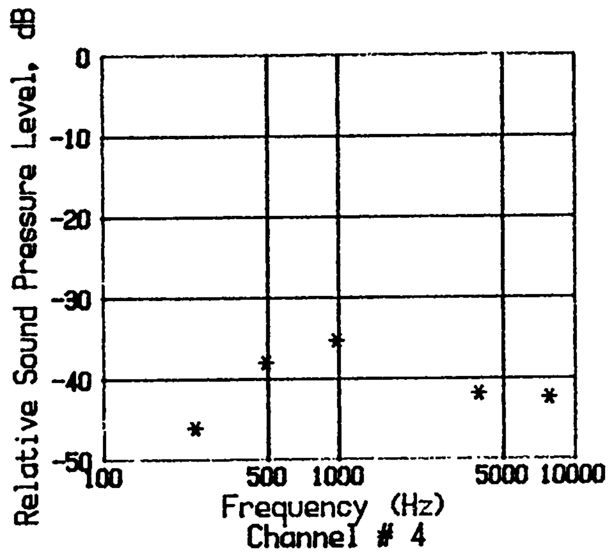


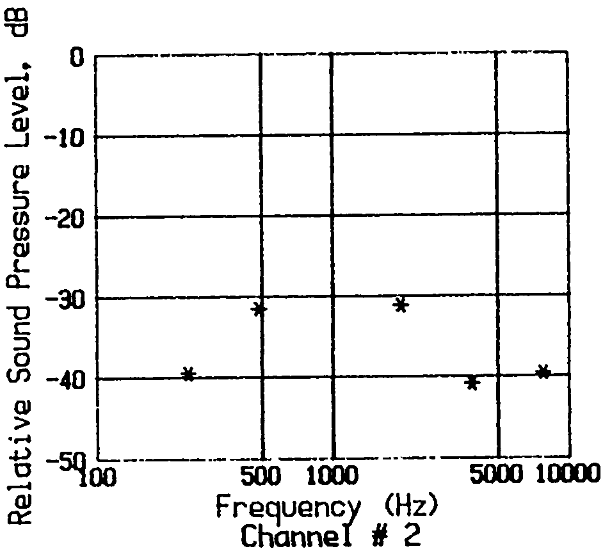
Flatville
June 20, 1984
Run 2.2



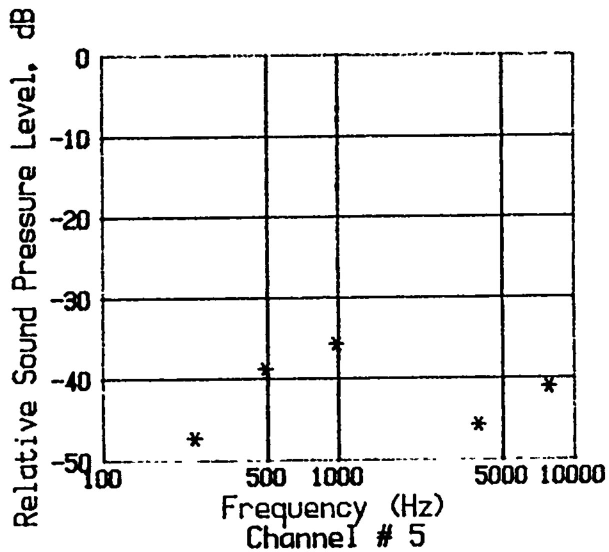


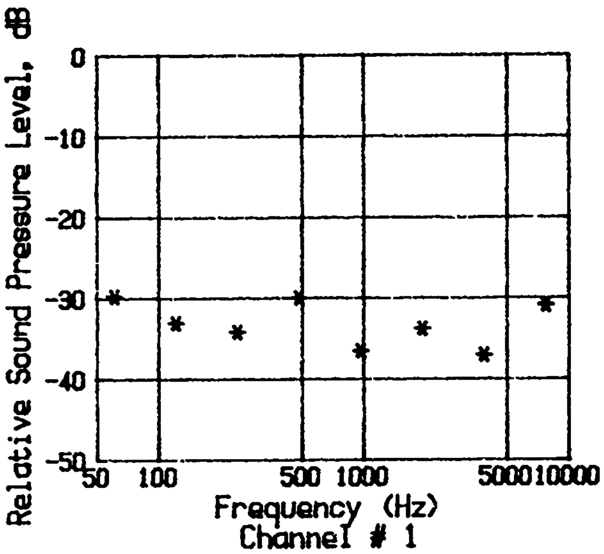


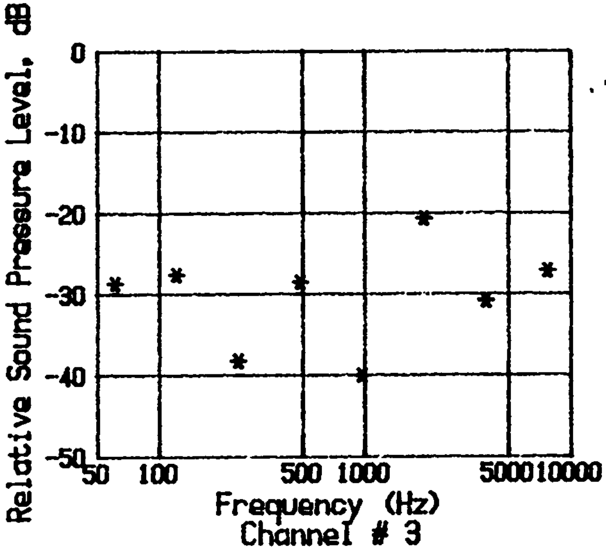


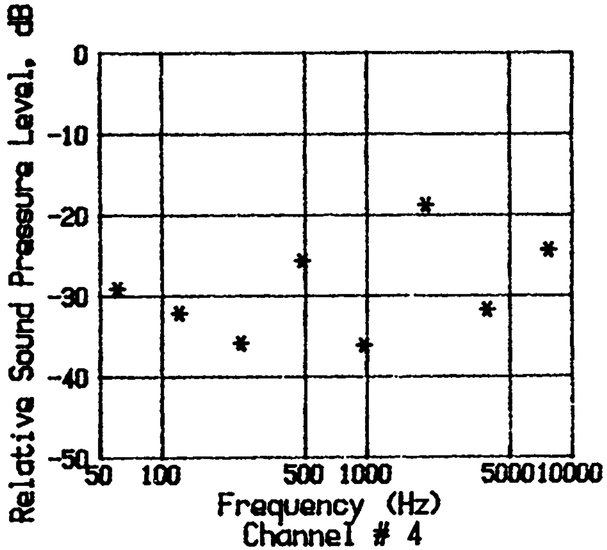


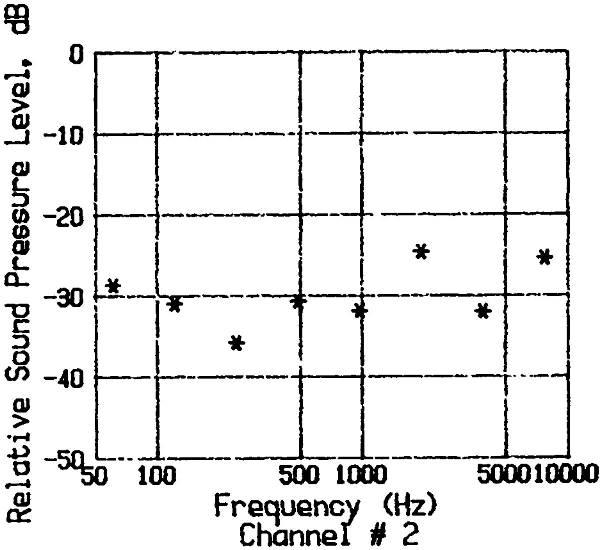
Flatville
June 20, 1984
Run 2.3



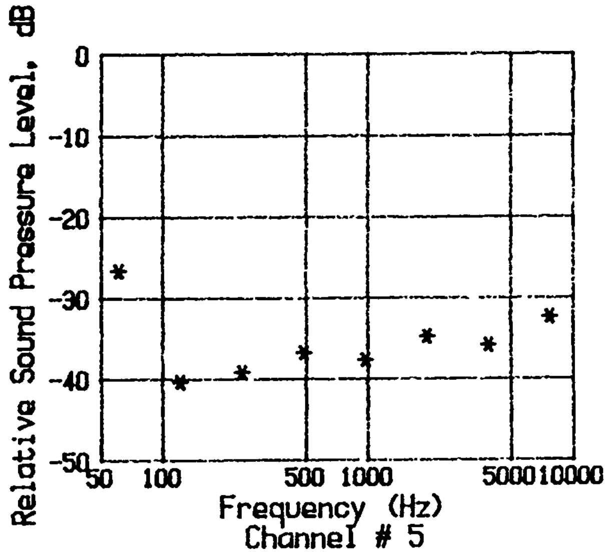


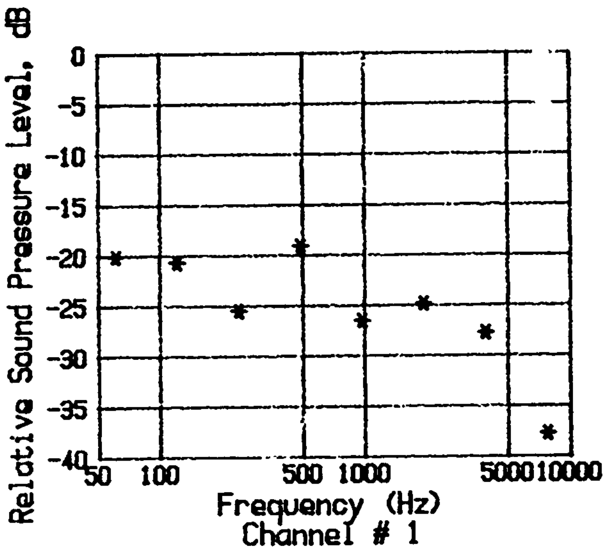


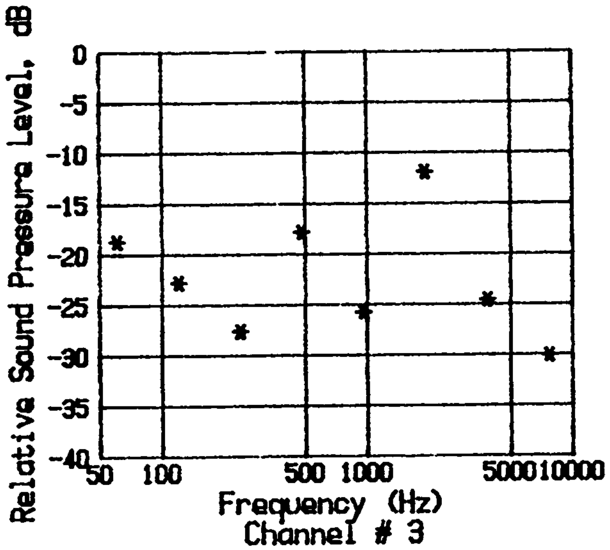


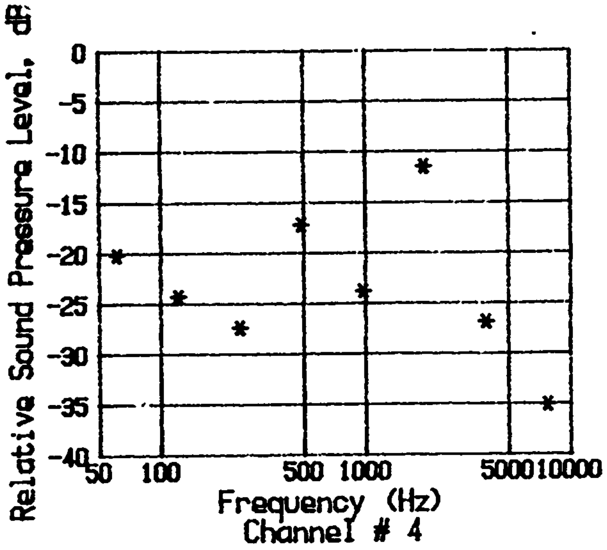


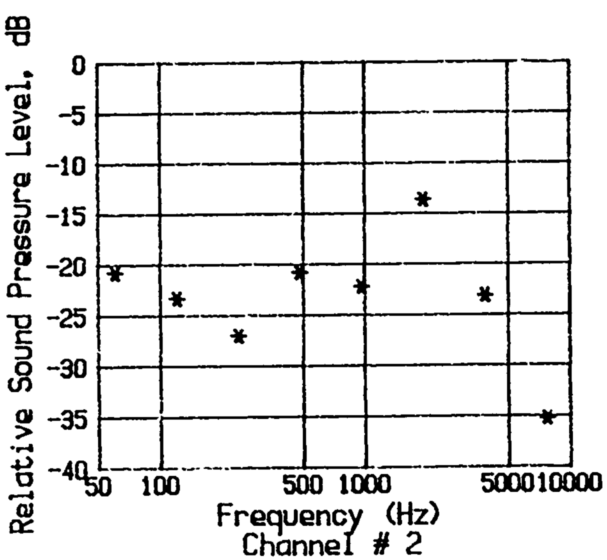
Bondville
Dec. 13, 1984
Run 1.1



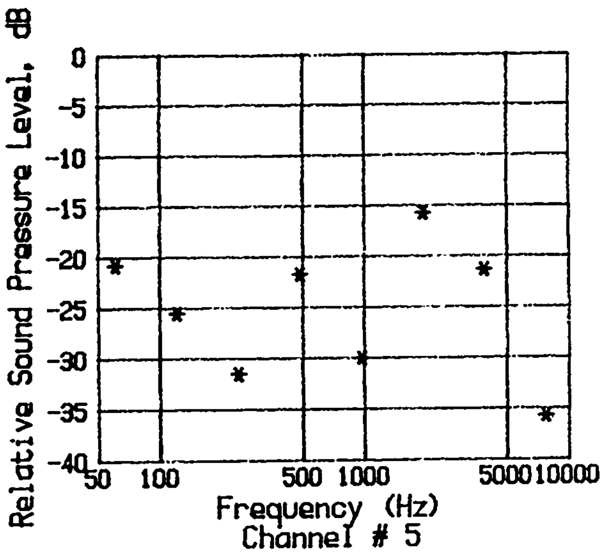


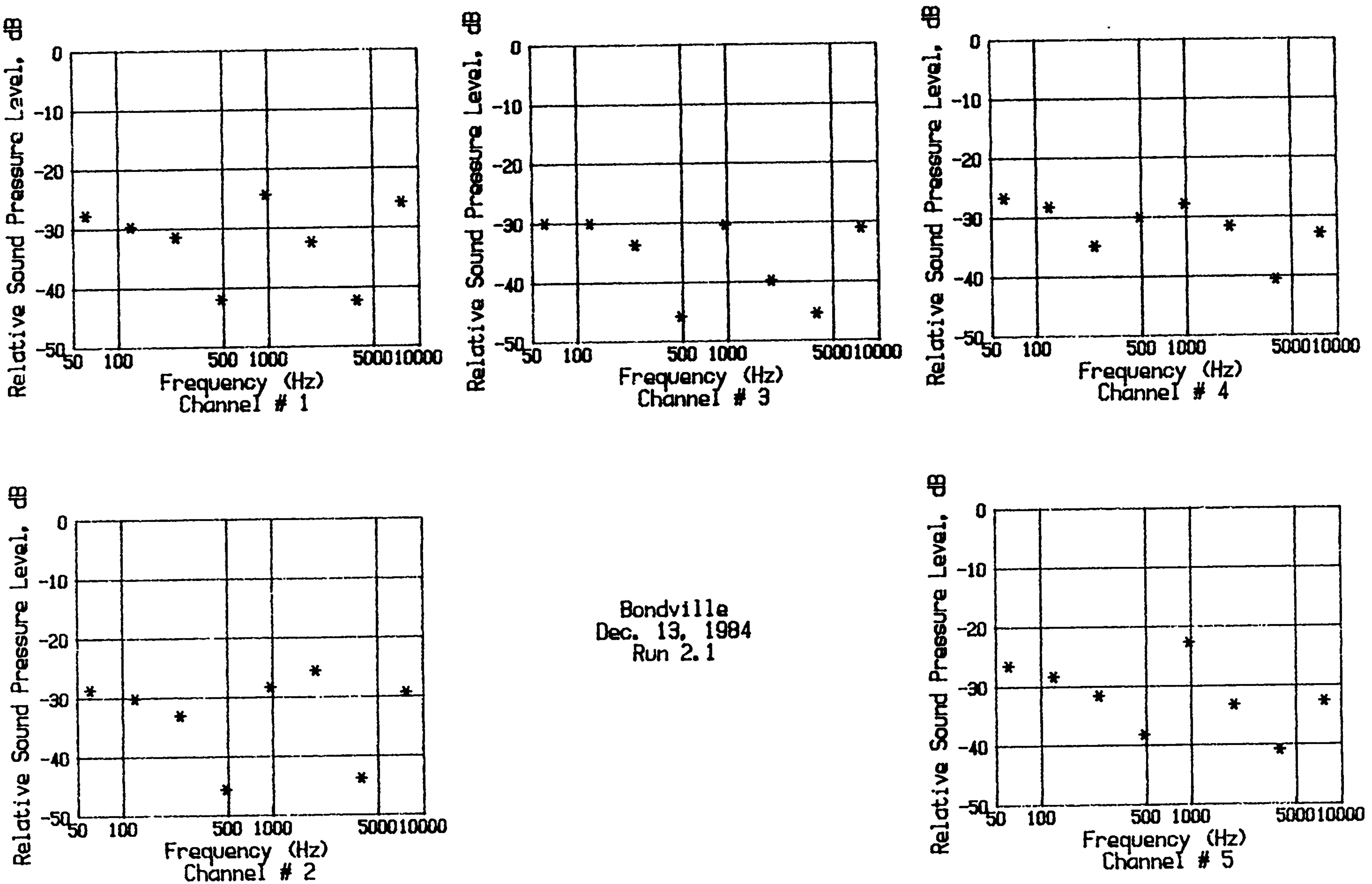


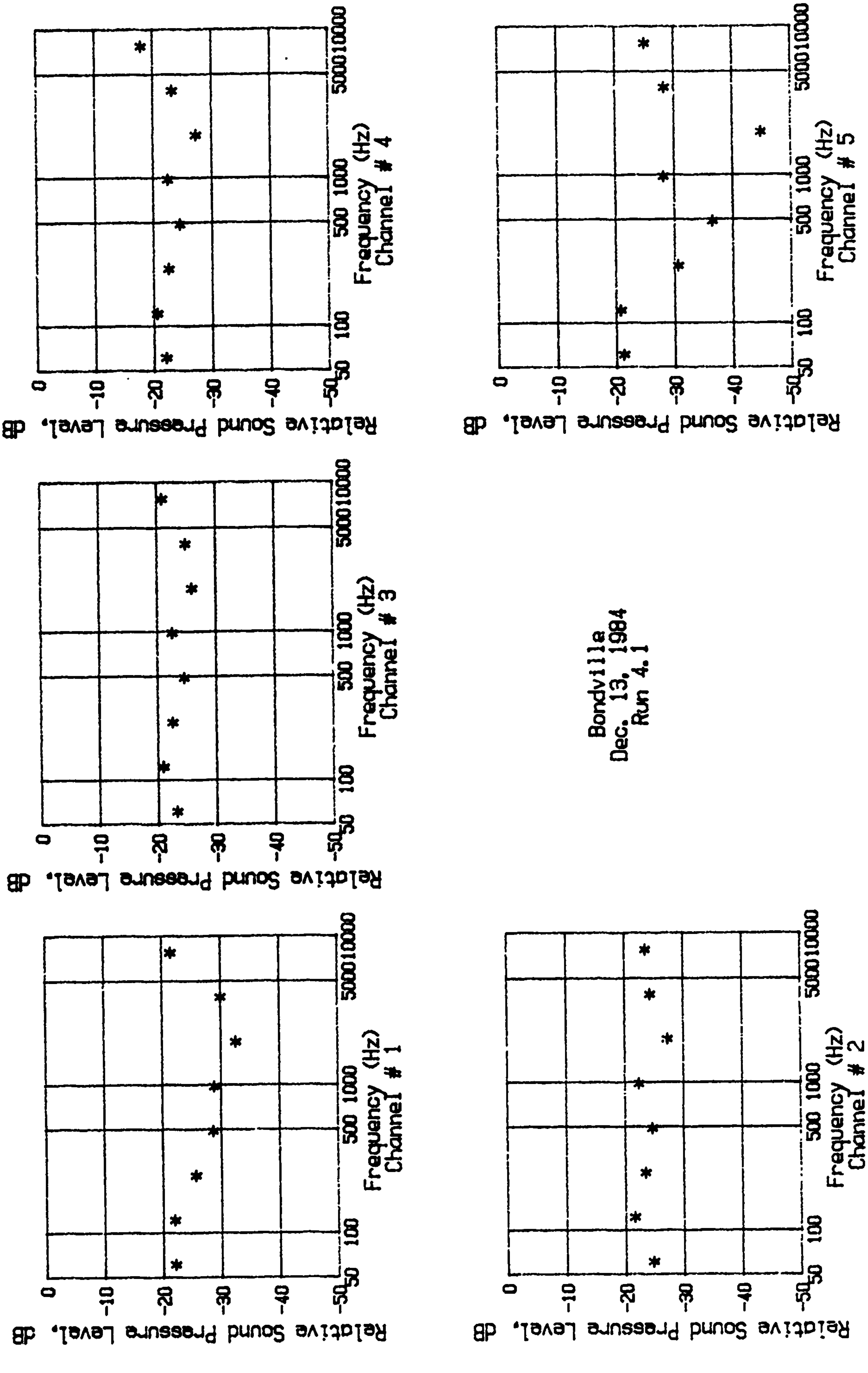




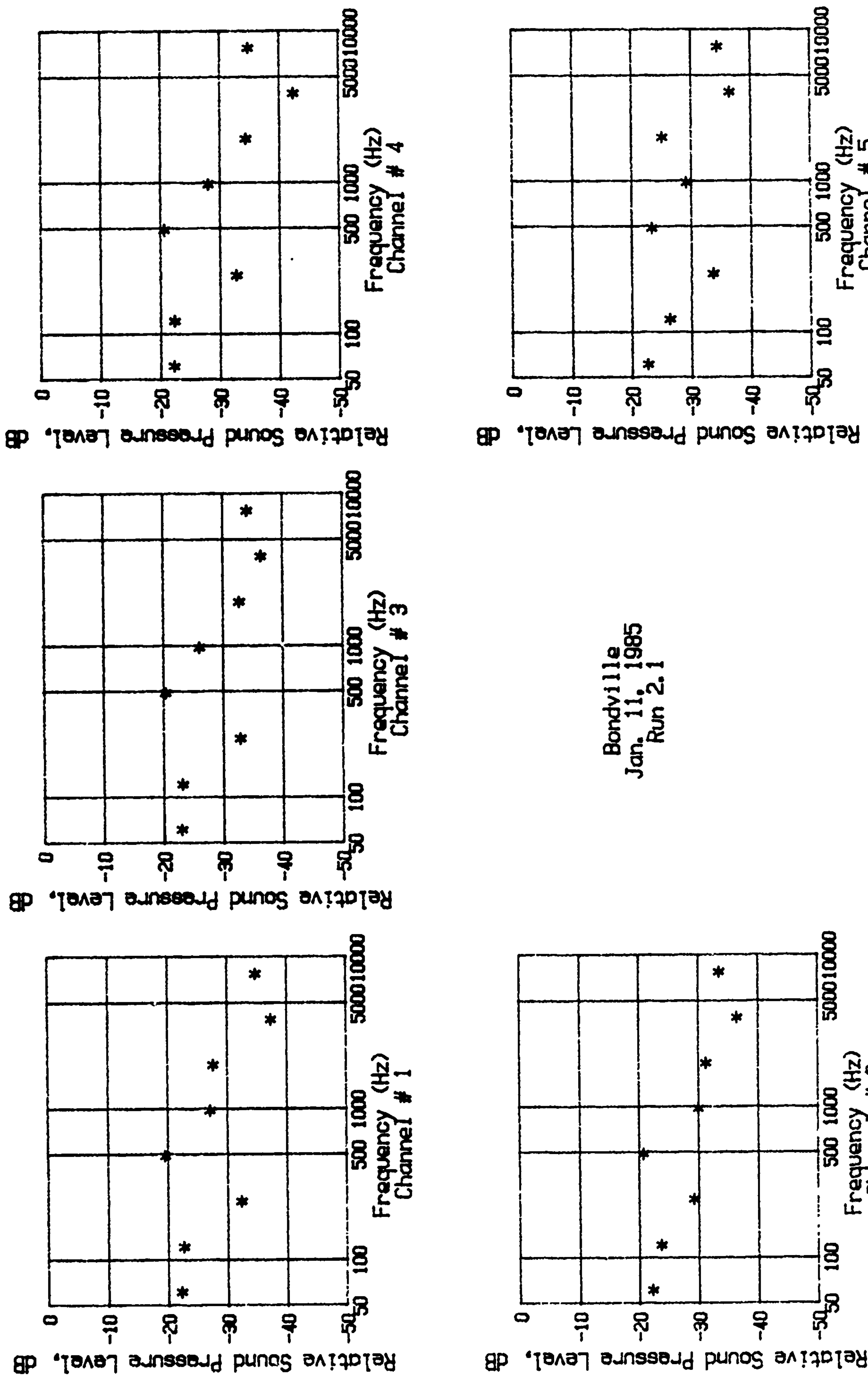
Bondville
Dec. 13, 1984
Run 1.2



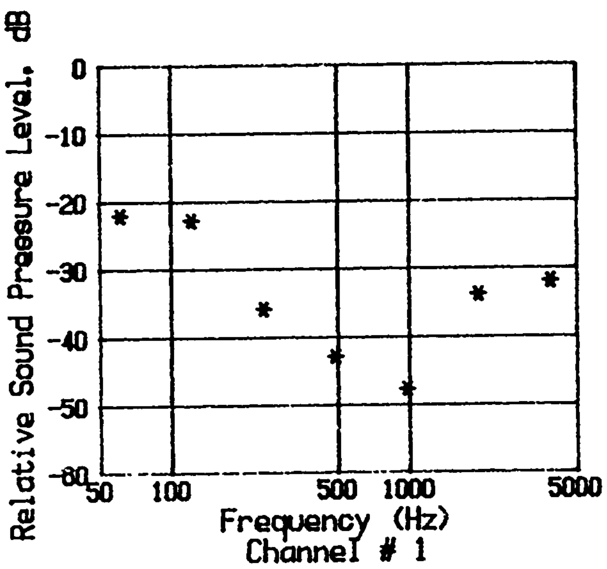

Bondville
Dec. 13, 1984
Run 2.1
Relative Sound Pressure Level, dB
Frequency (Hz)
Channel # 1
Channel # 2
Channel # 3
Channel # 4
Channel # 5

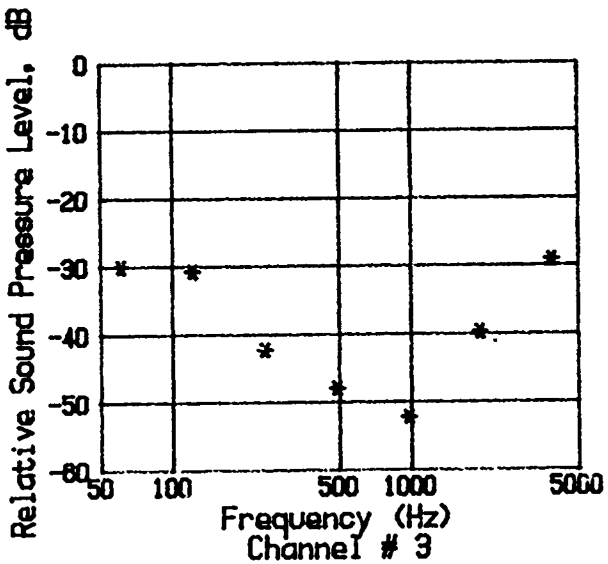


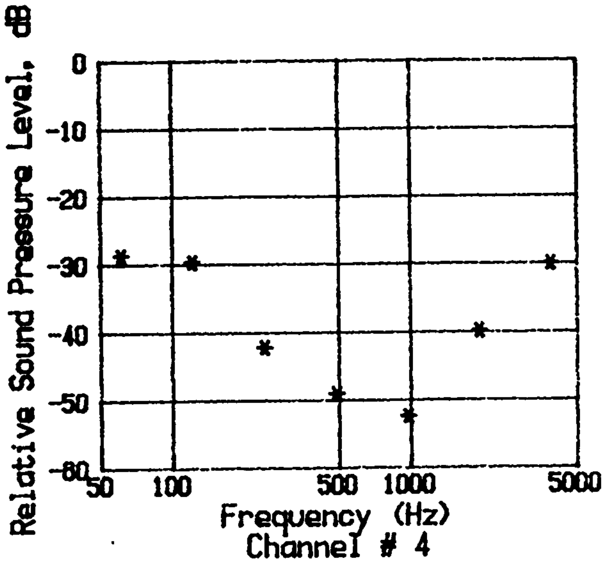
Relative Sound Pressure Level, dB
Frequency (Hz)
Channel # 1
Channel # 2
Channel # 3
Channel # 4
Channel # 5
Bondville
Dec. 13, 1984
Run 4.1

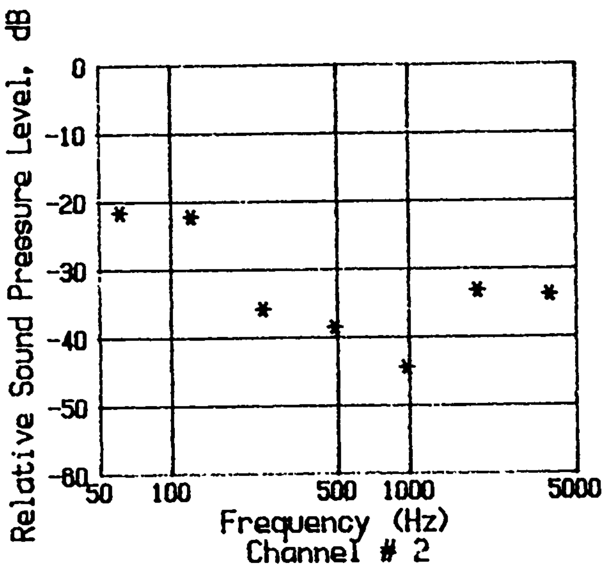


Relative Sound Pressure Level, dB
0
-10
-20
-30
-40
-50
50 100 500 1000 5000 10000
Frequency (Hz)
Channel # 1
Channel # 2
Channel # 3
Channel # 4
Channel # 5
Bondville
Jan. 11, 1985
Run 2.1

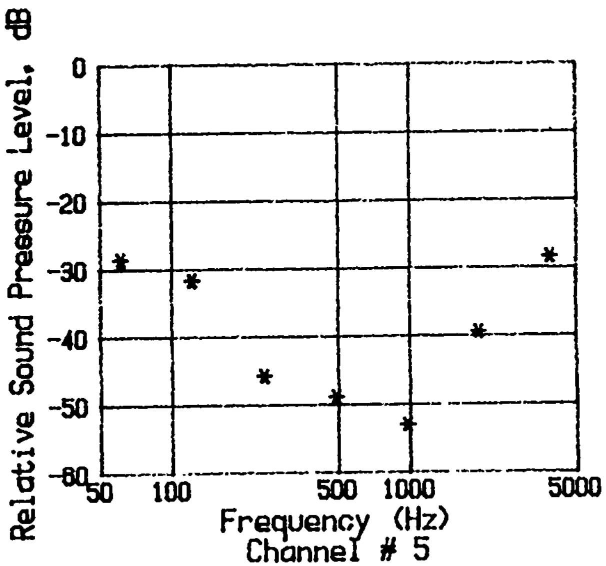


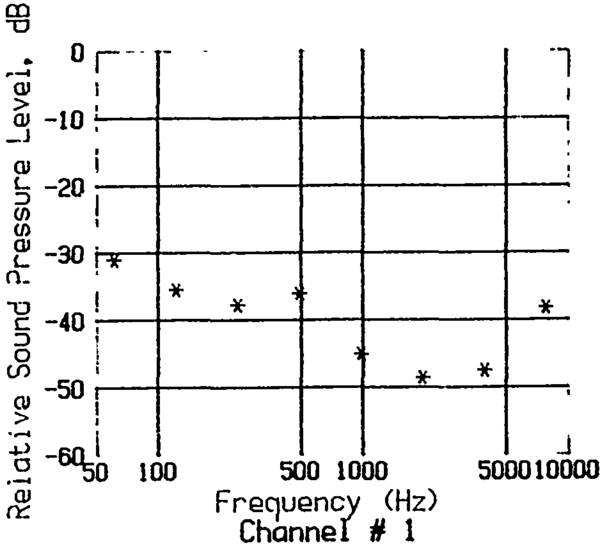


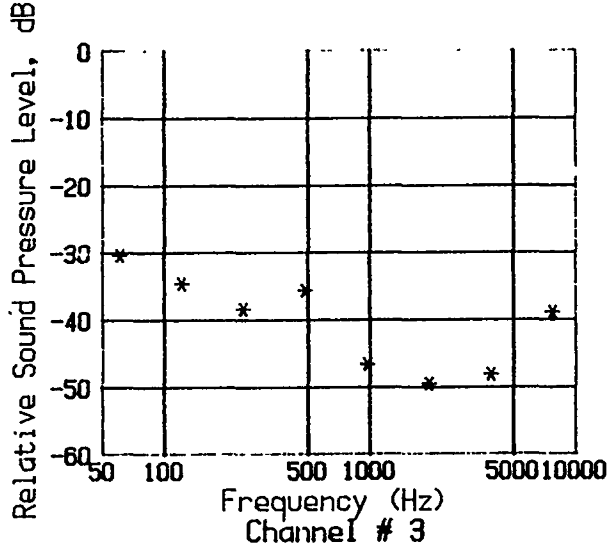


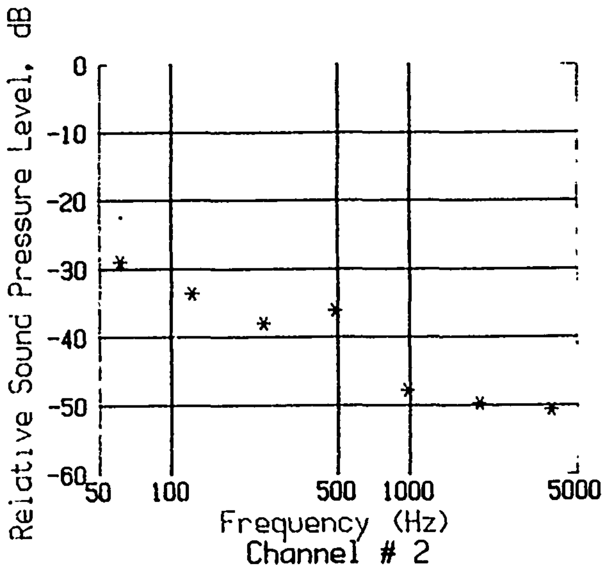


Bondville
Jan. 11, 1985
Run 2.2

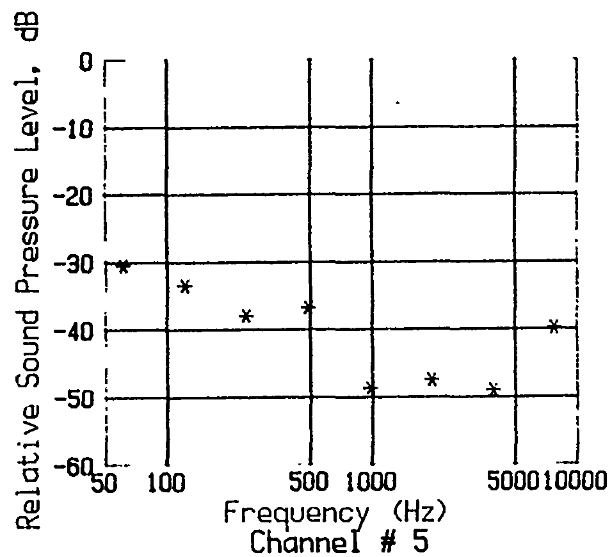


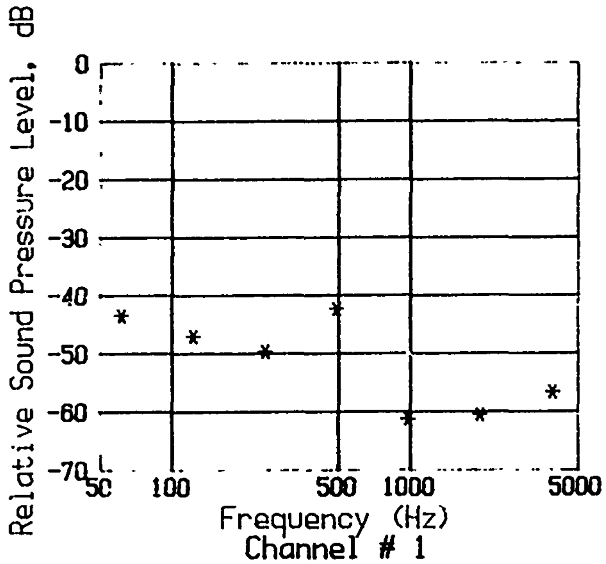


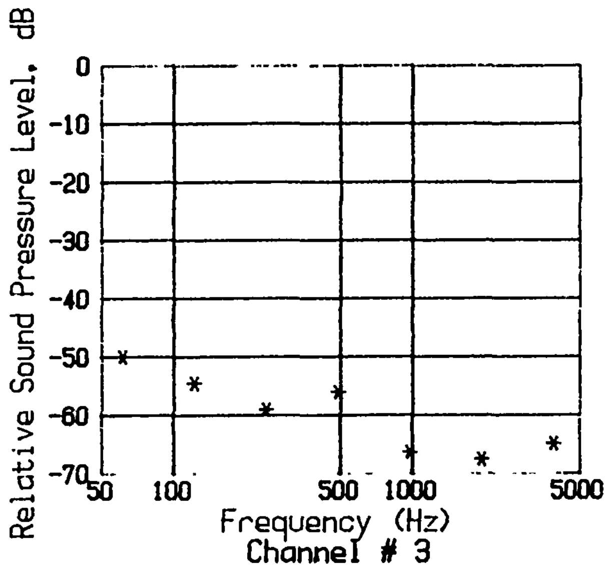


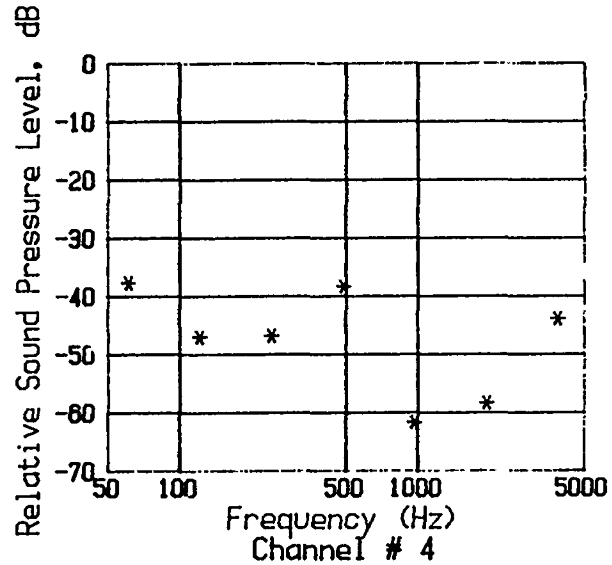


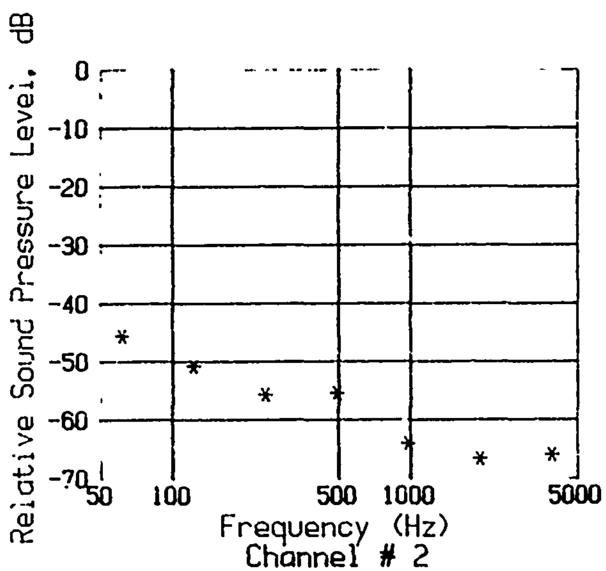
Bondville
July 23, 1985
Run 1



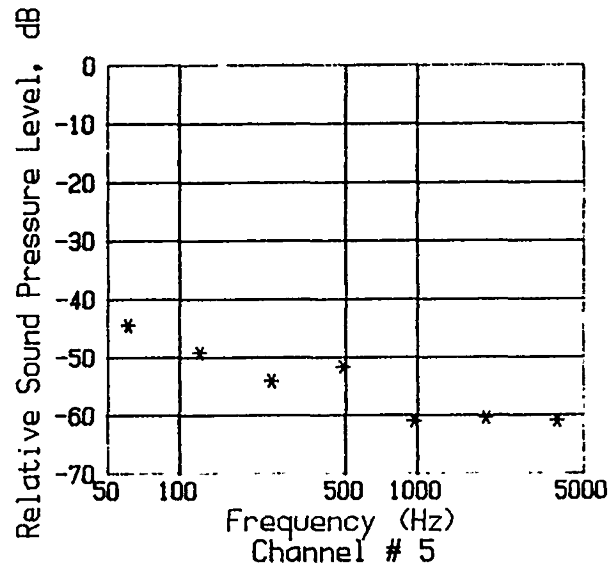


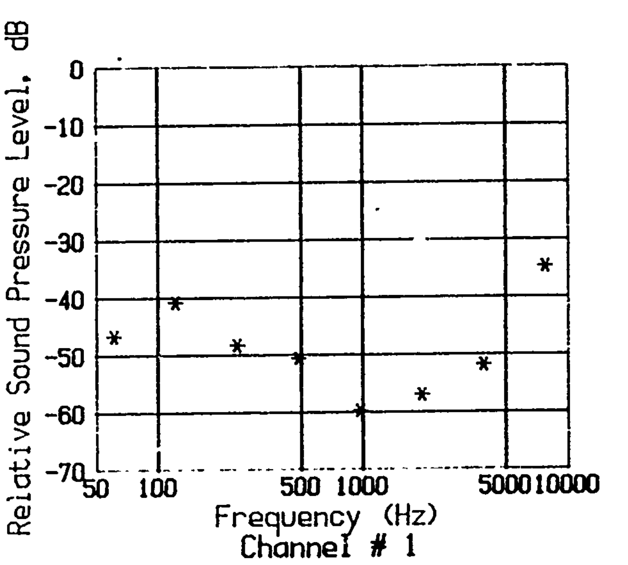


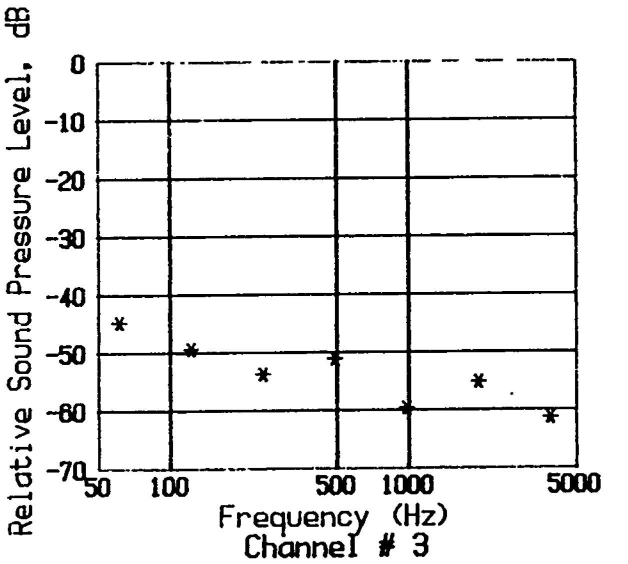


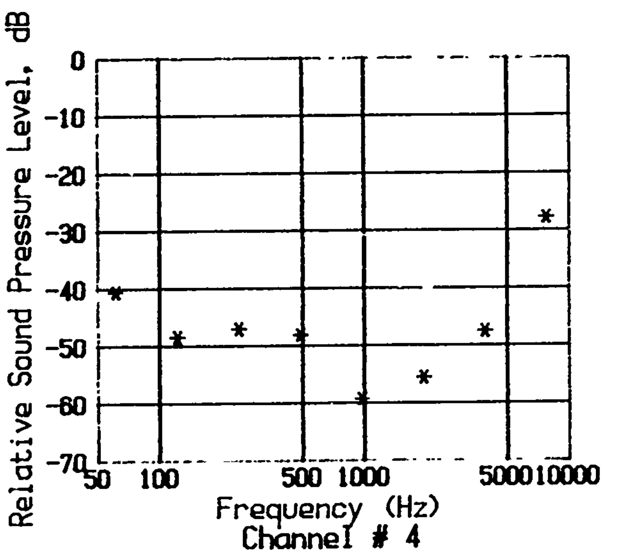


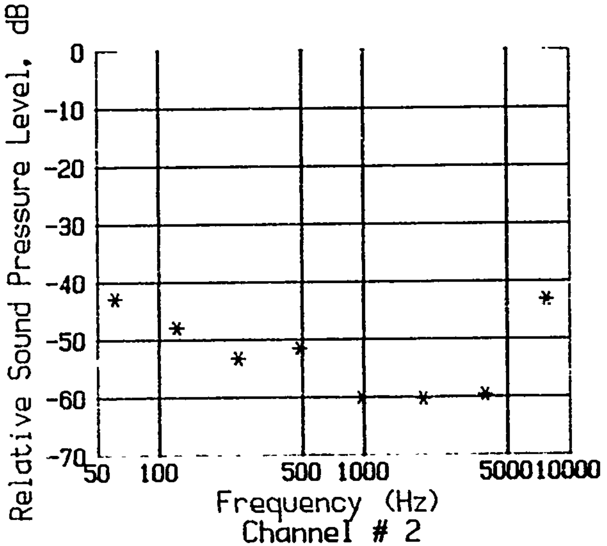
Bondville
July 23, 1985
Run 2



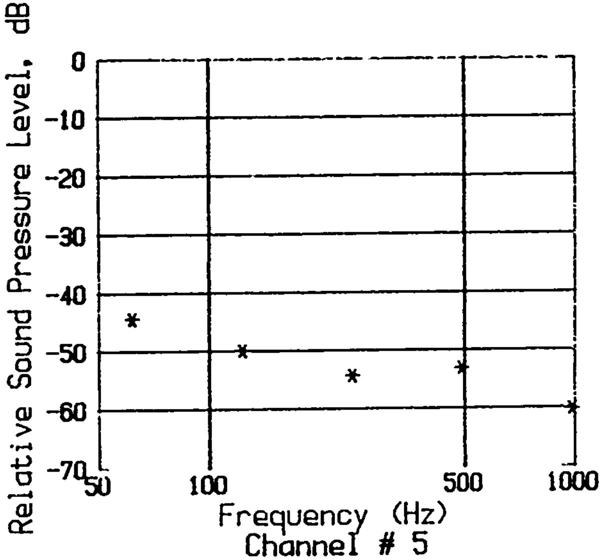


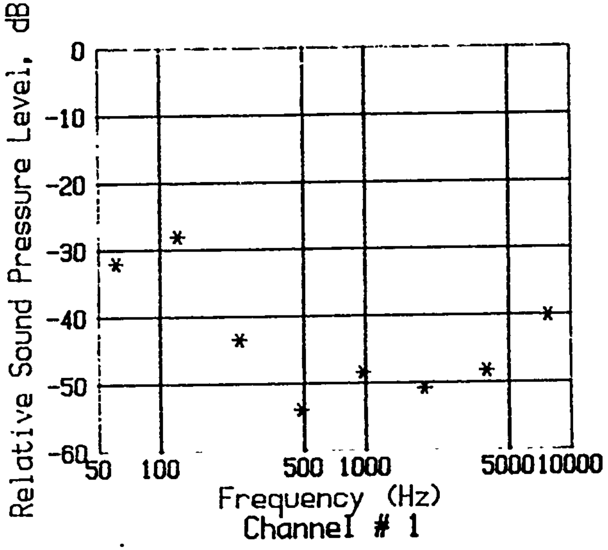


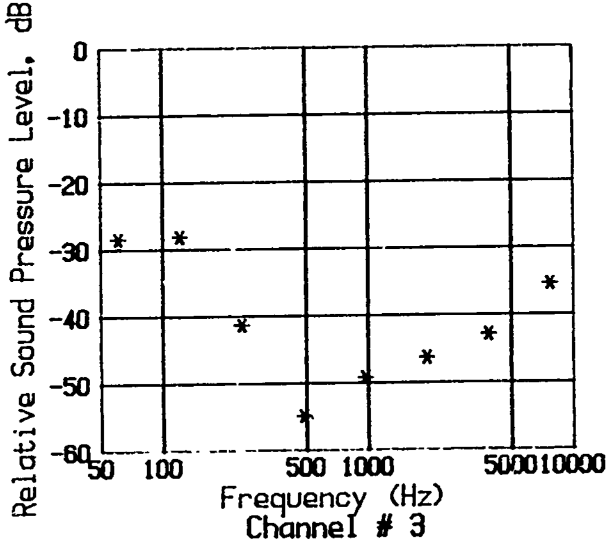


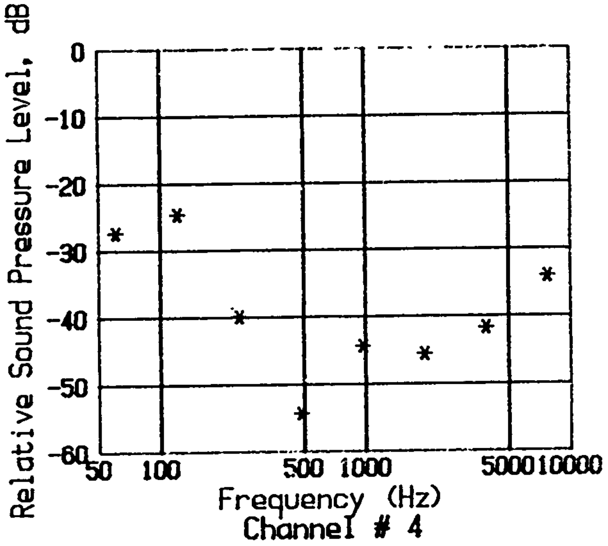


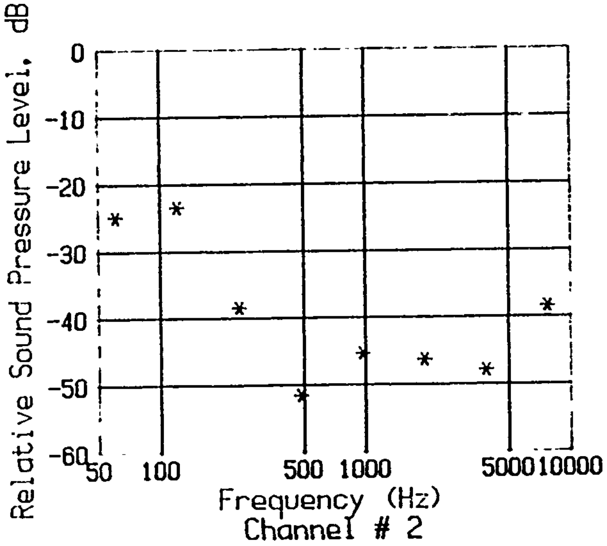
Bondville
July 23, 1985
Run 3



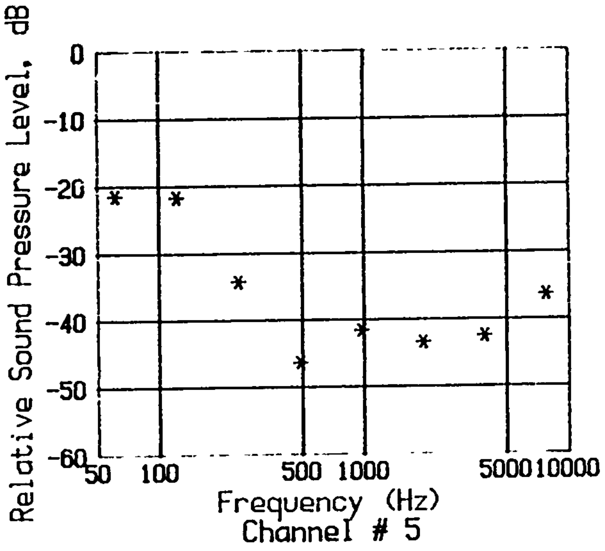


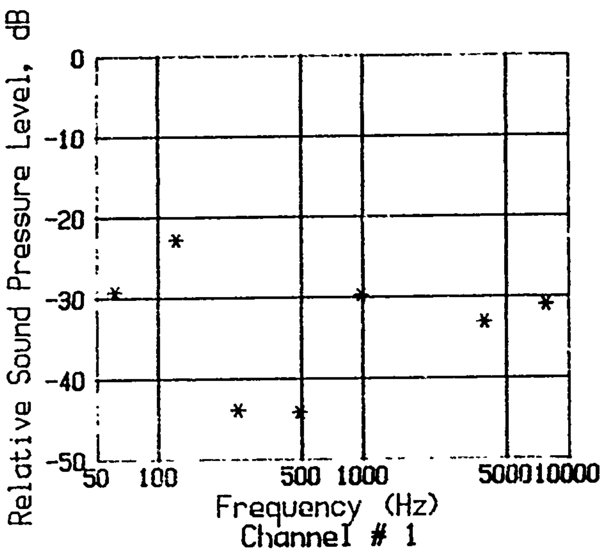


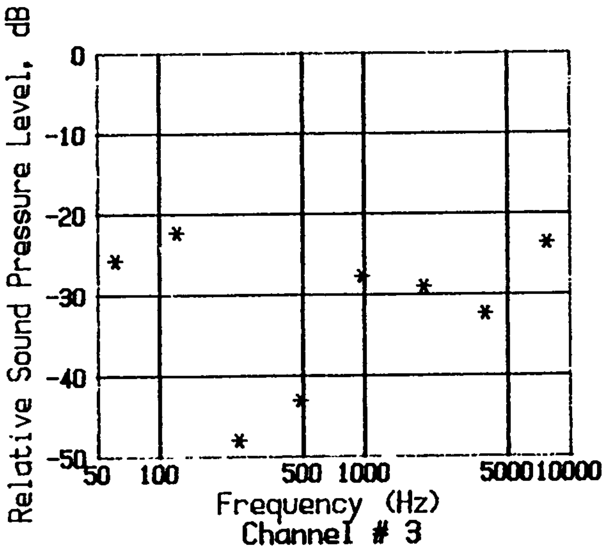


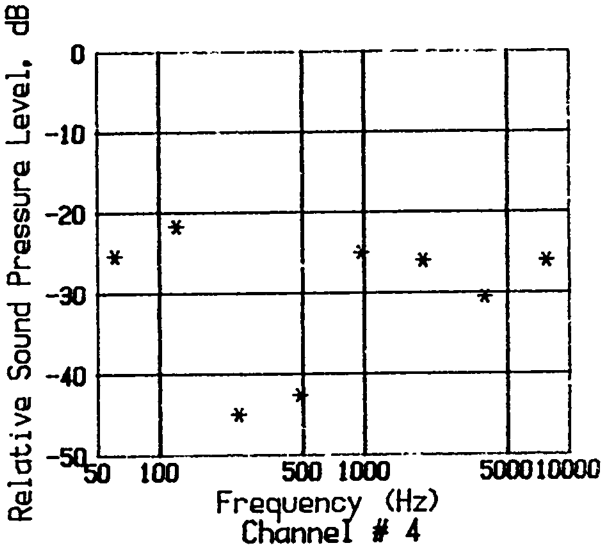


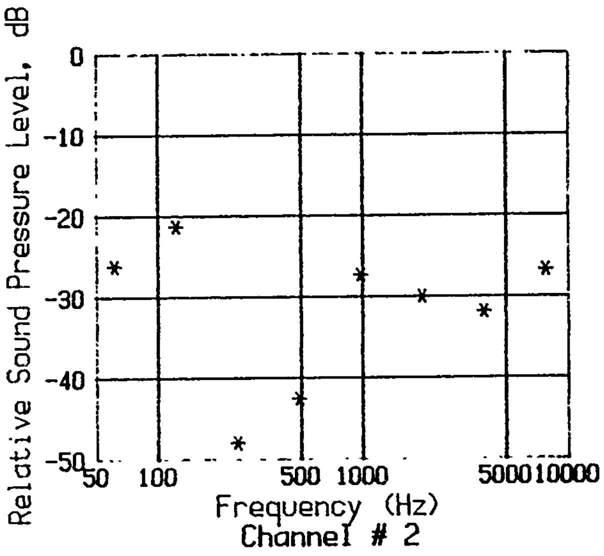
Sandusky
July 25, 1985
Run 1



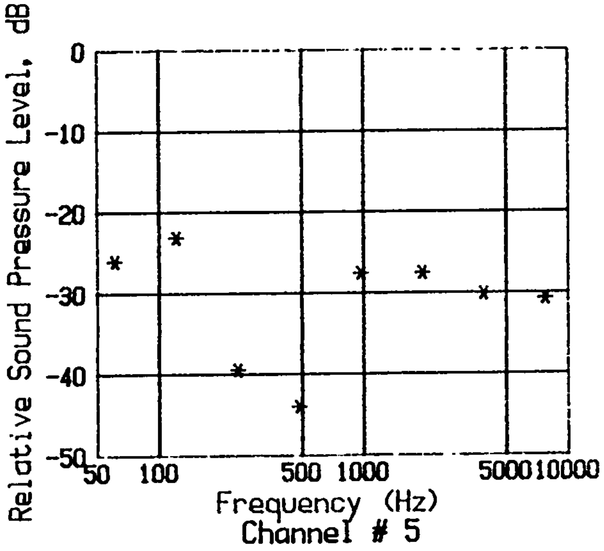


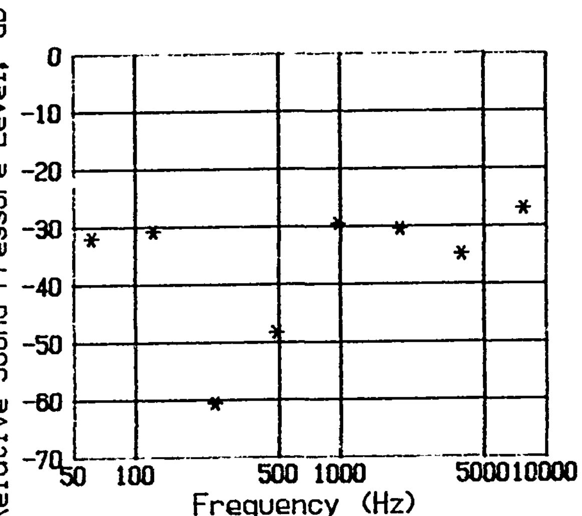




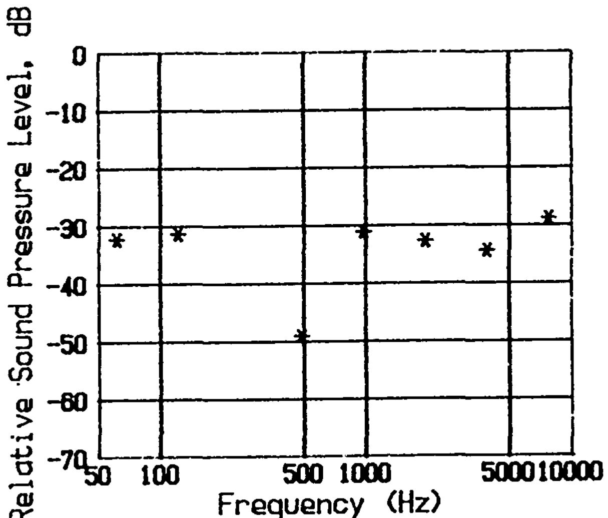


Sandusky
July 25, 1985
Run 2

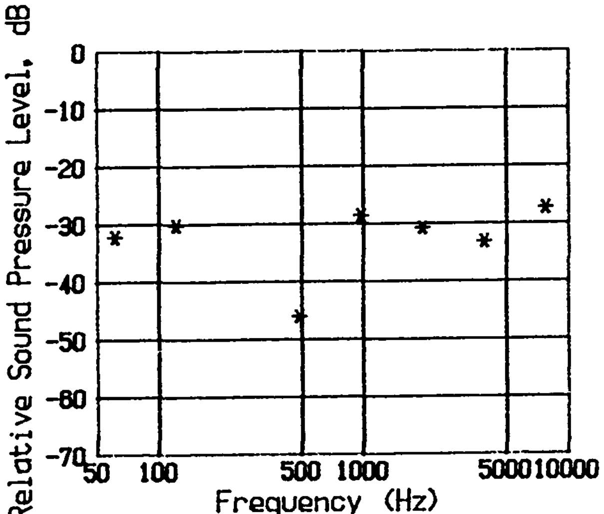




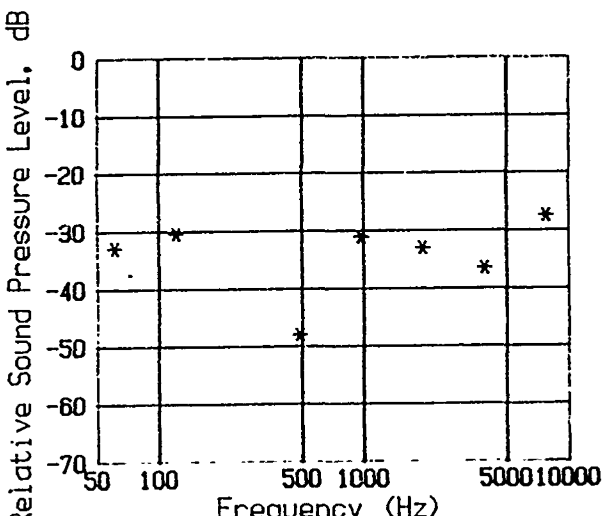
Channel # 1



Channel # 3

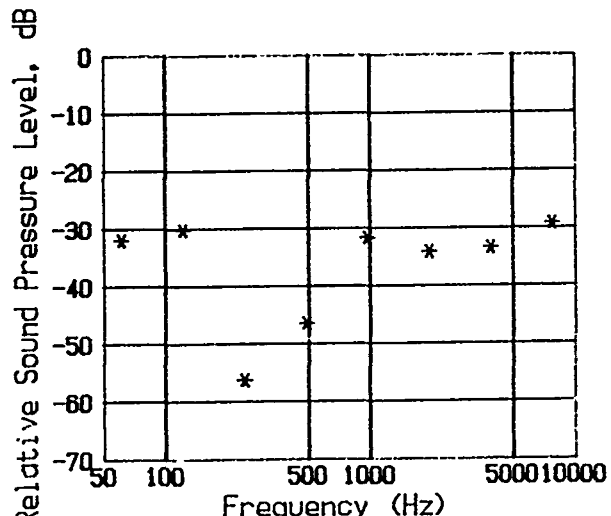


Channel # 4



Channel # 2

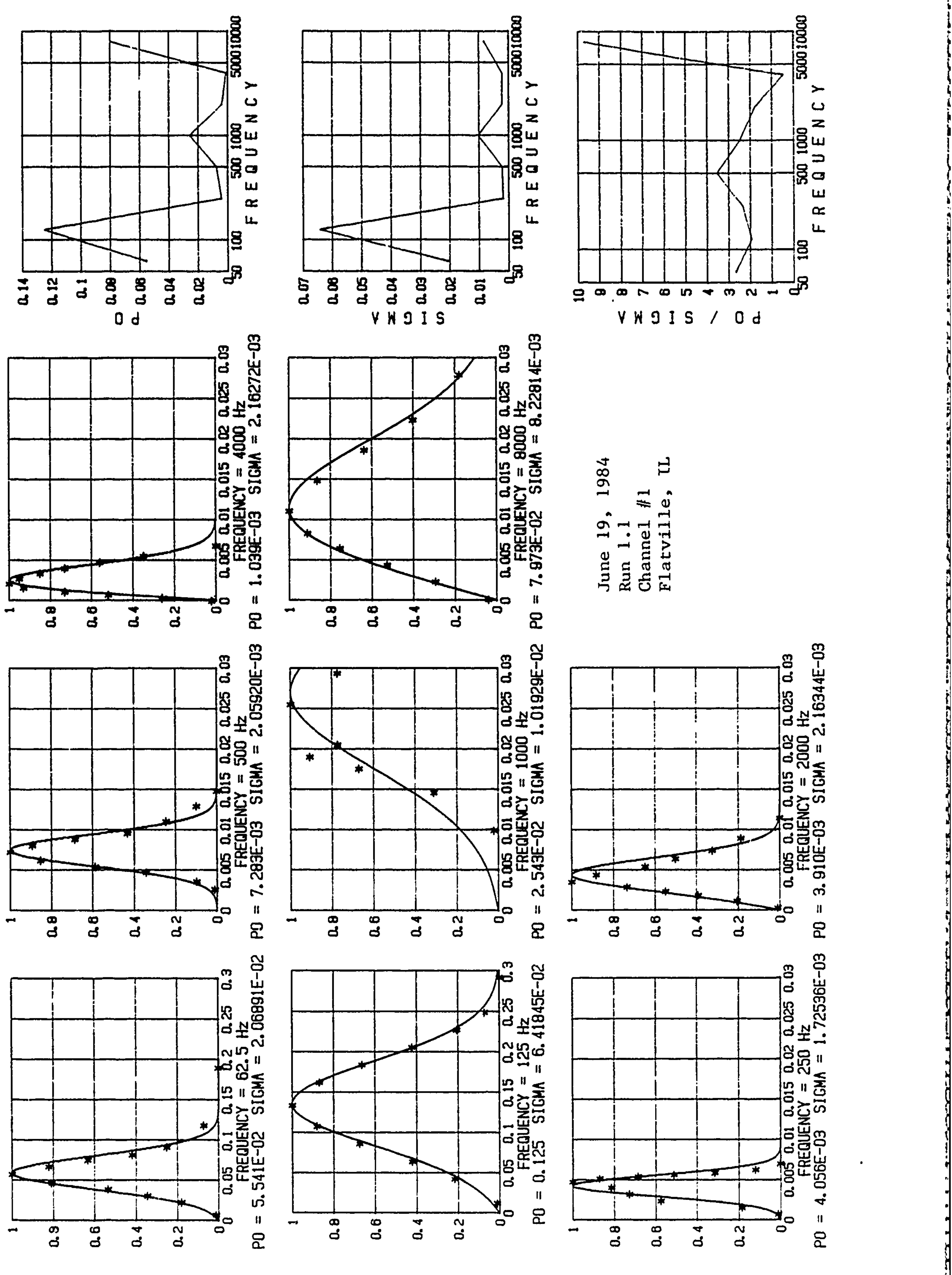
Sandusky
July 25, 1985
Run 3



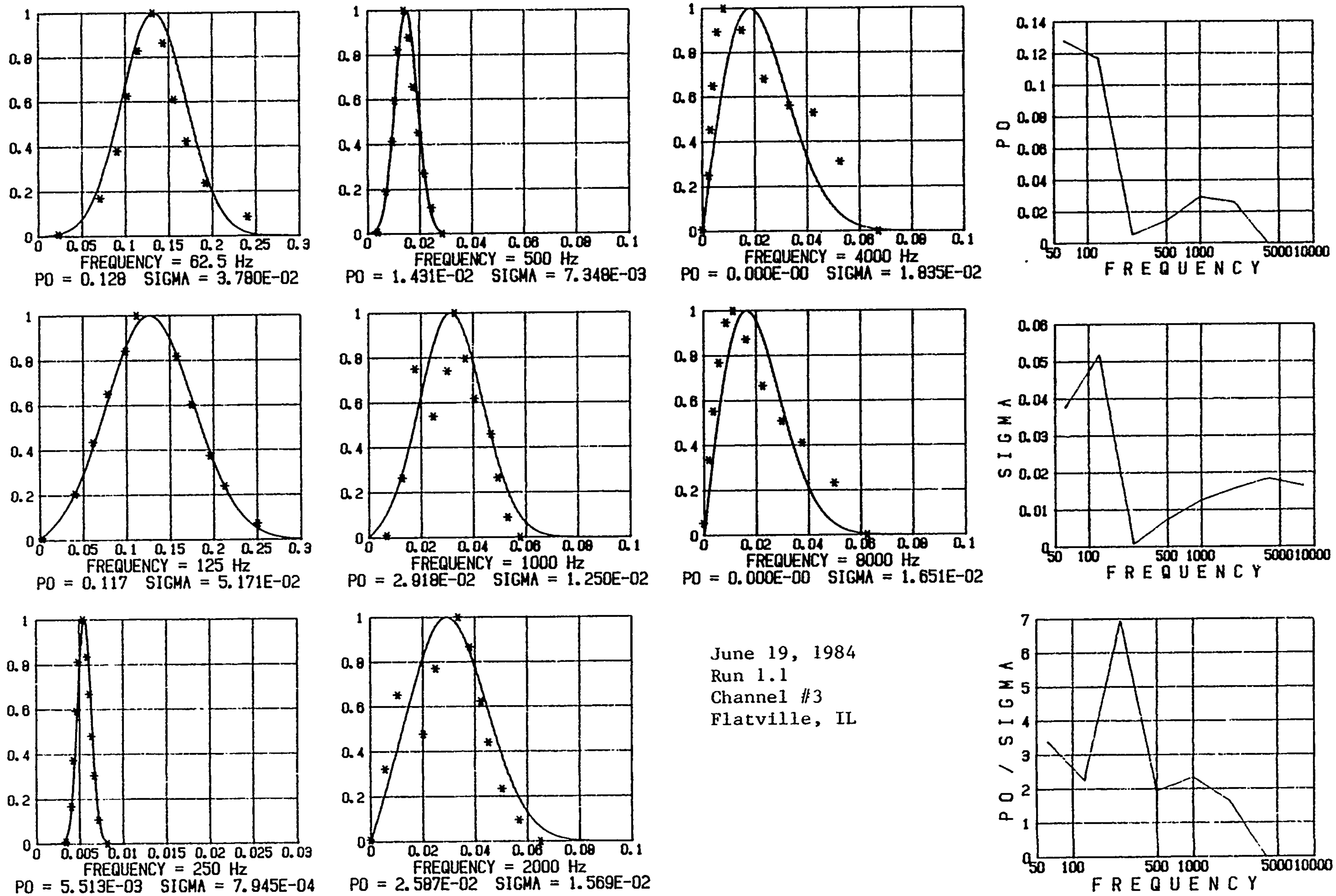
Channel # 5

APPENDIX E

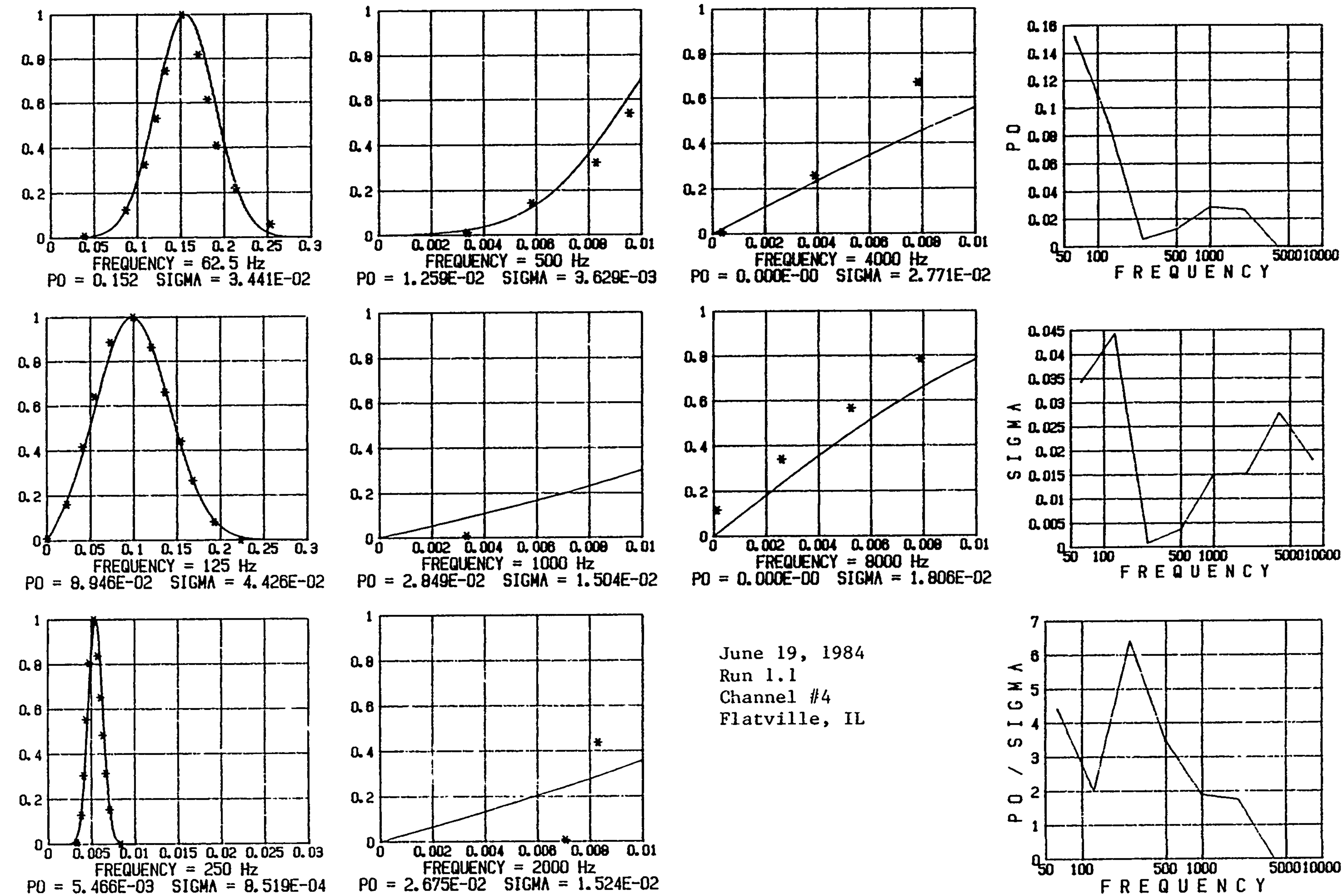
Comparison of the MCA data with the bivariant normal probability function.



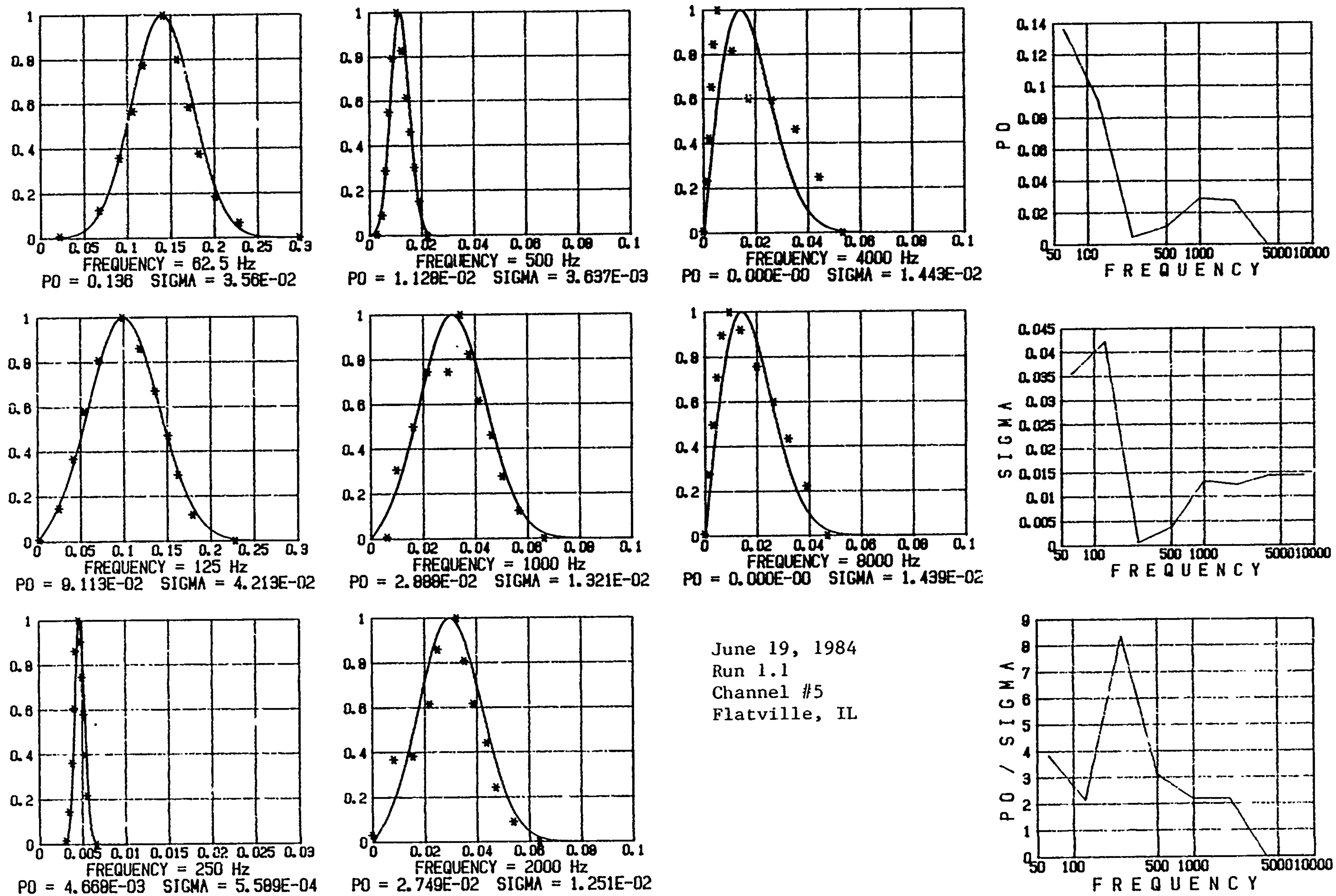
FREQUENCY = 62.5 Hz
PO = 5.541E-02 SIGMA = 2.06891E-02
FREQUENCY = 125 Hz
PO = 0.125 SIGMA = 6.41845E-02
FREQUENCY = 250 Hz
PO = 4.056E-03 SIGMA = 1.72536E-03
FREQUENCY = 500 Hz
PO = 7.283E-03 SIGMA = 2.05920E-03
FREQUENCY = 1000 Hz
PO = 2.543E-02 SIGMA = 1.01929E-02
FREQUENCY = 2000 Hz
PO = 3.910E-03 SIGMA = 2.16344E-03
FREQUENCY = 4000 Hz
PO = 1.039E-03 SIGMA = 2.16272E-03
FREQUENCY = 8000 Hz
PO = 7.973E-02 SIGMA = 8.22814E-03
June 19, 1984
Run 1.1
Channel #1
Flatville, IL
P O
F R E Q U E N C Y
S I G M A
P O / S I G M A



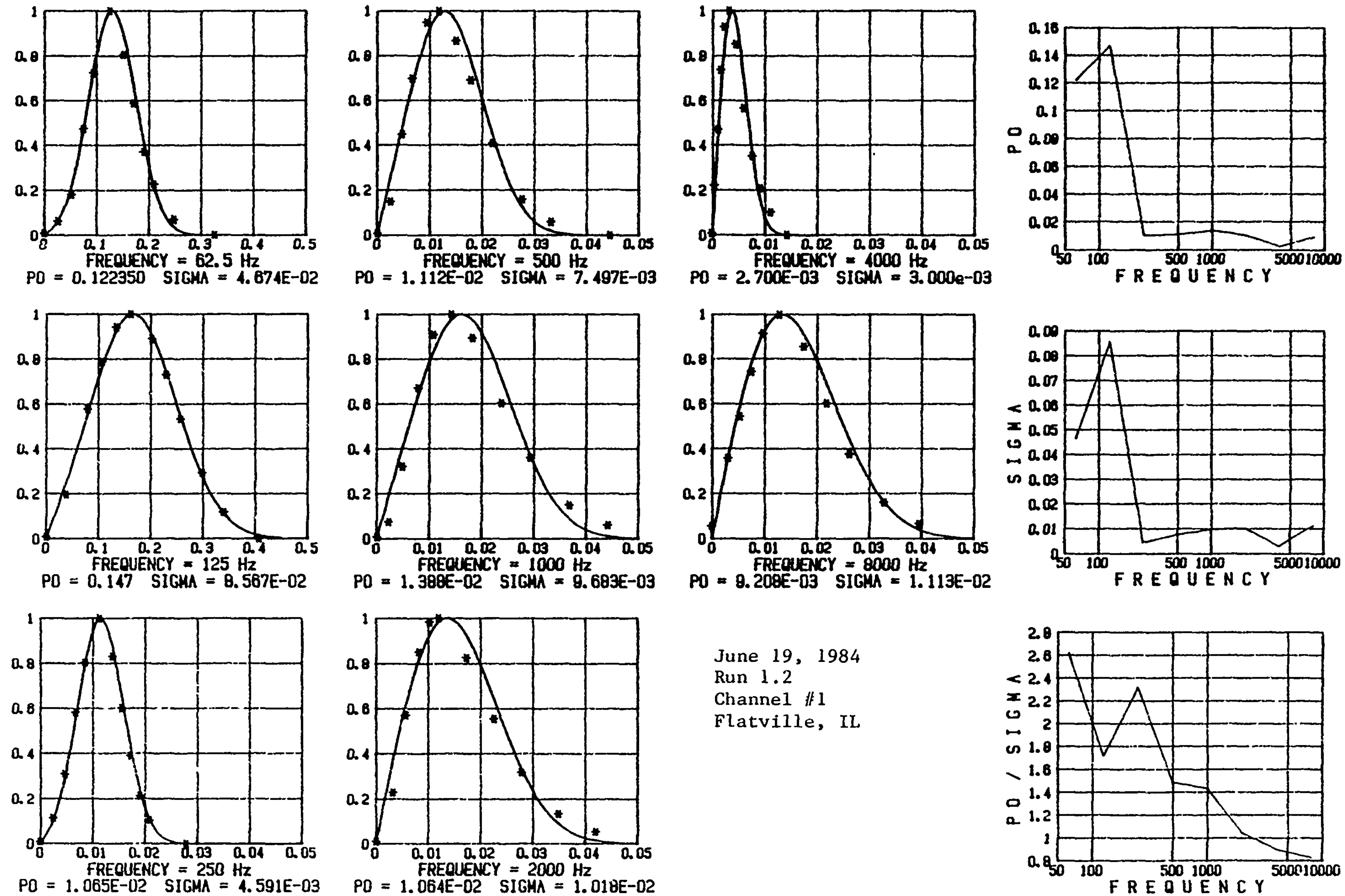
June 19, 1984
Run 1.1
Channel #3
Flatville, IL



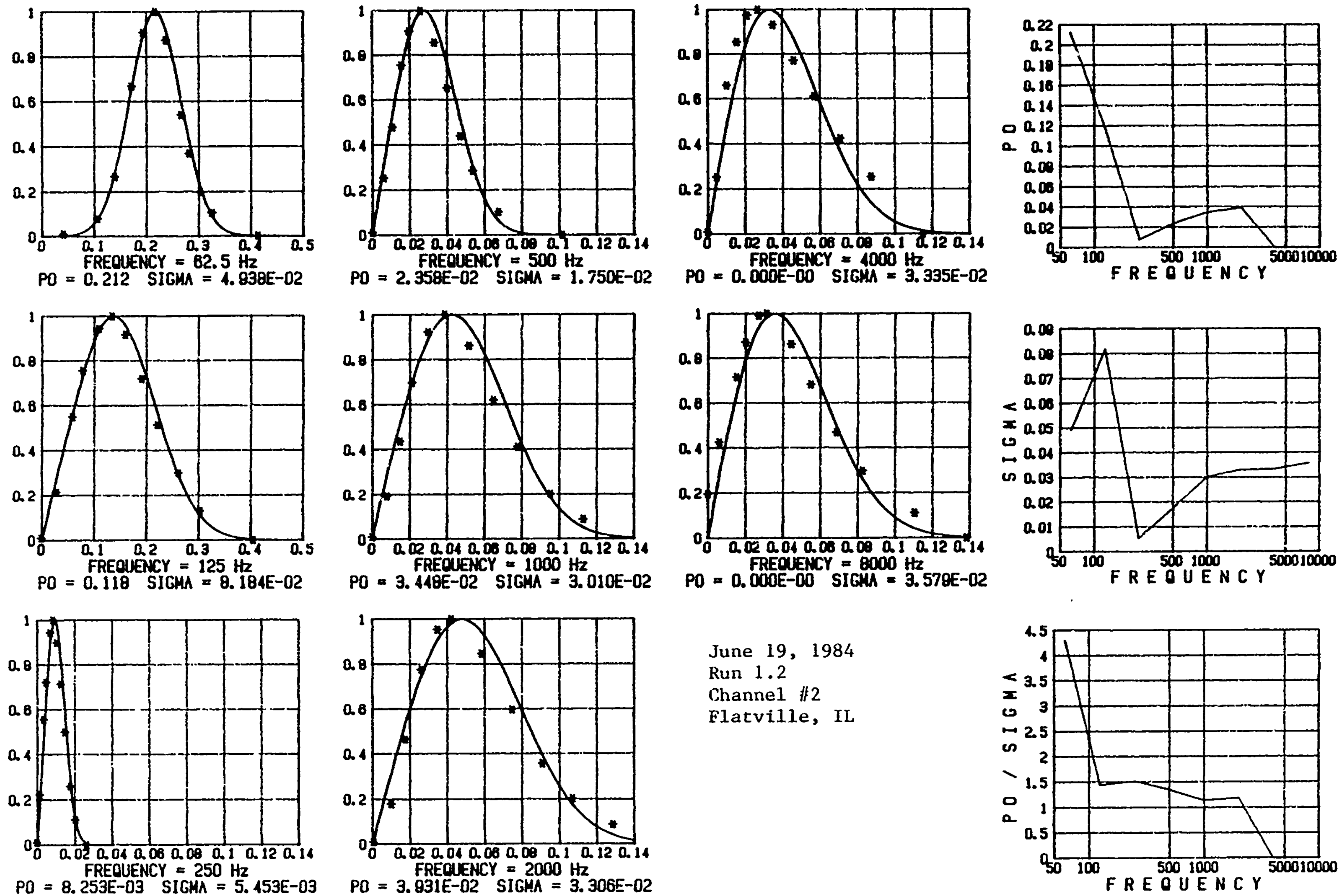
June 19, 1984
Run 1.1
Channel #4
Flatville, IL



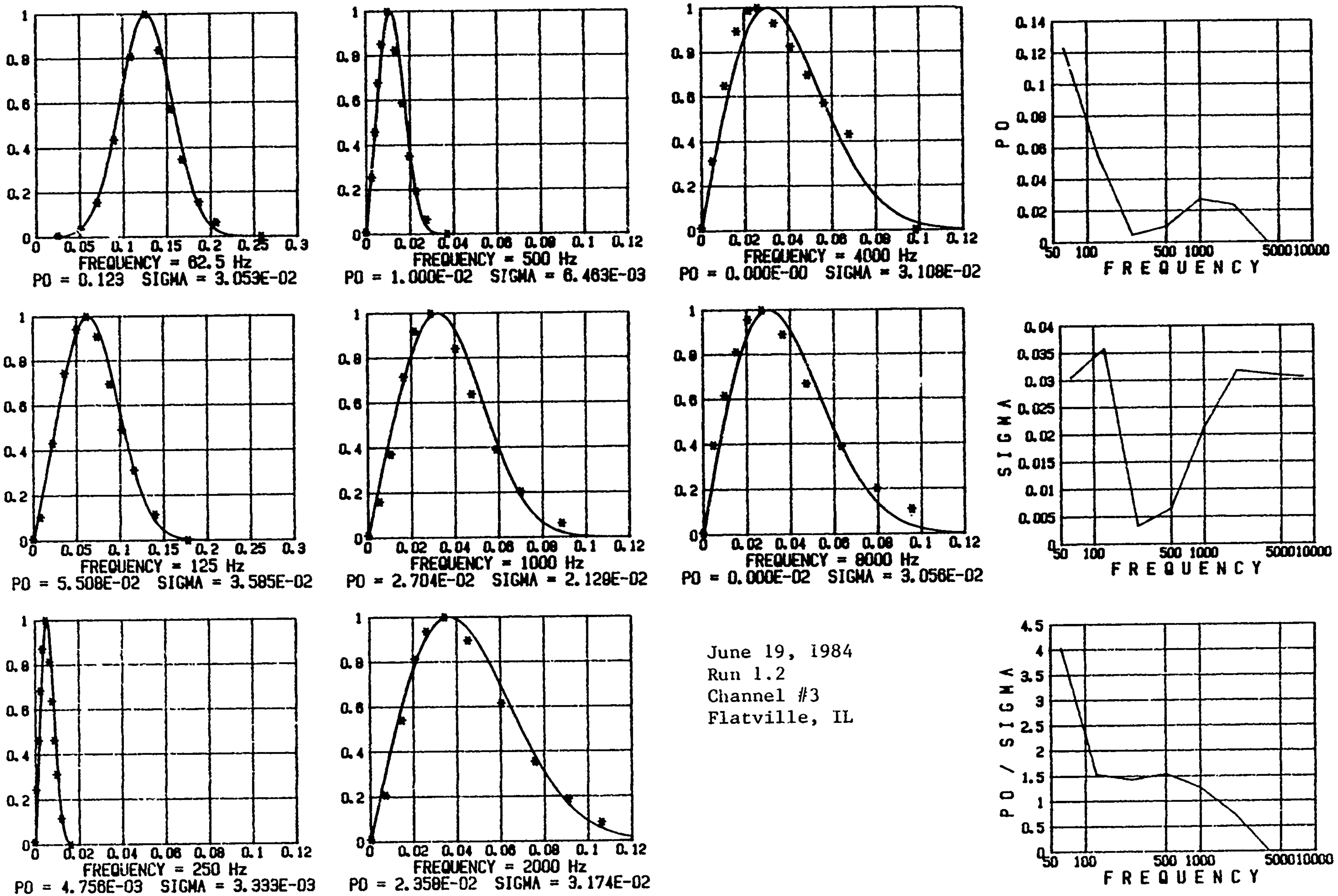
June 19, 1984
Run 1.1
Channel #5
Flatville, IL



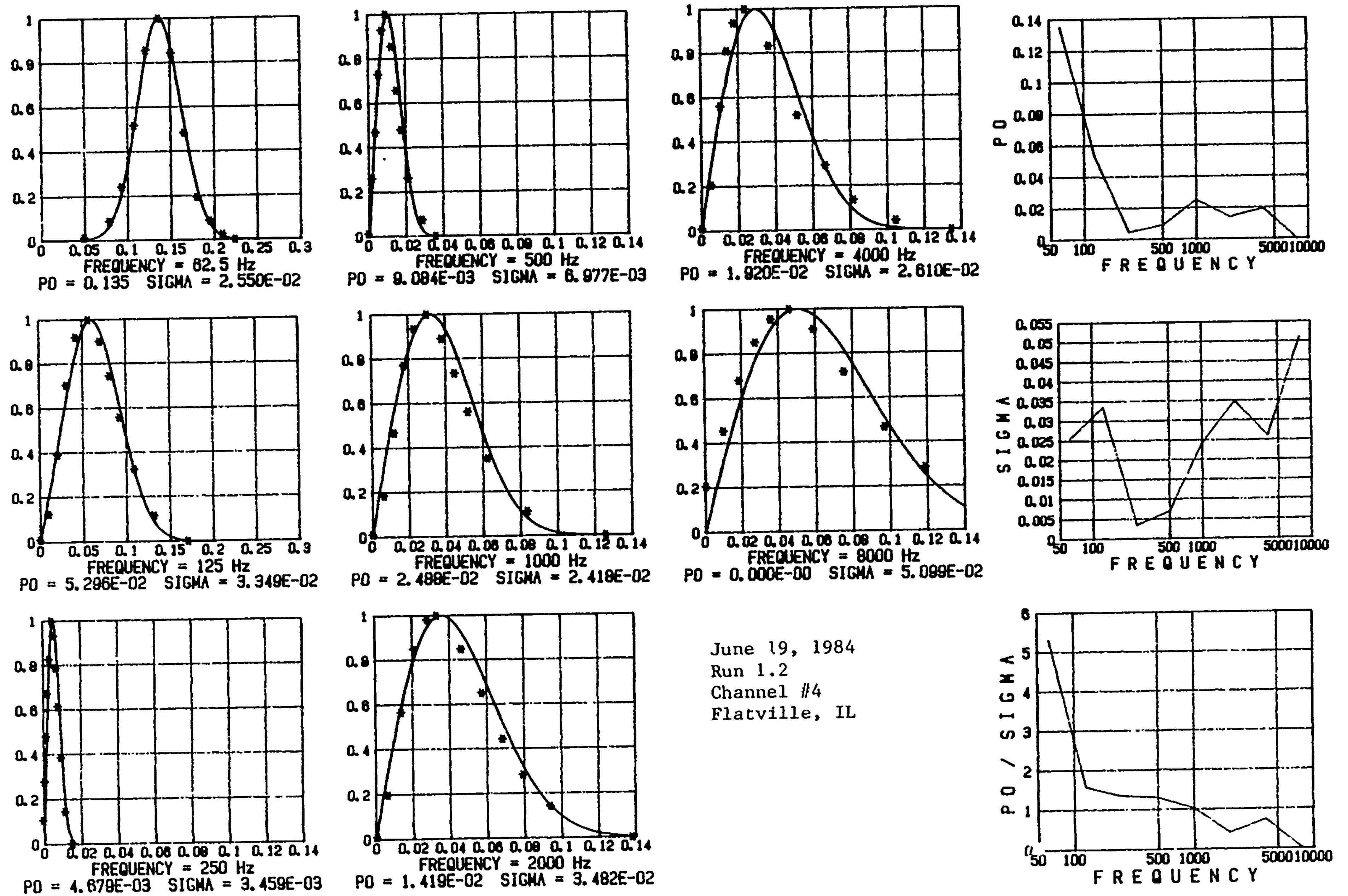
June 19, 1984
Run 1.2
Channel #1
Flatville, IL



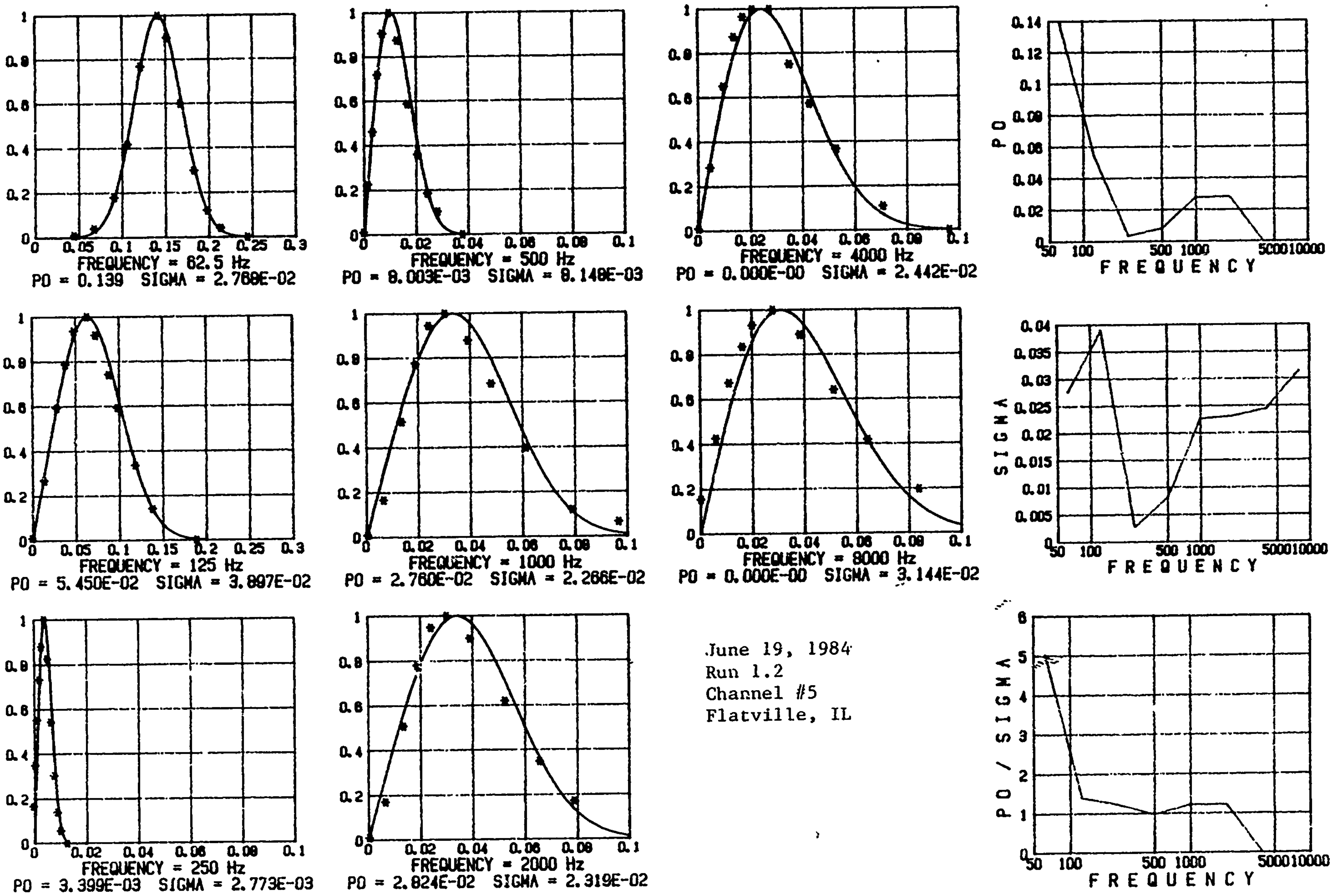
June 19, 1984
Run 1.2
Channel #2
Flatville, IL



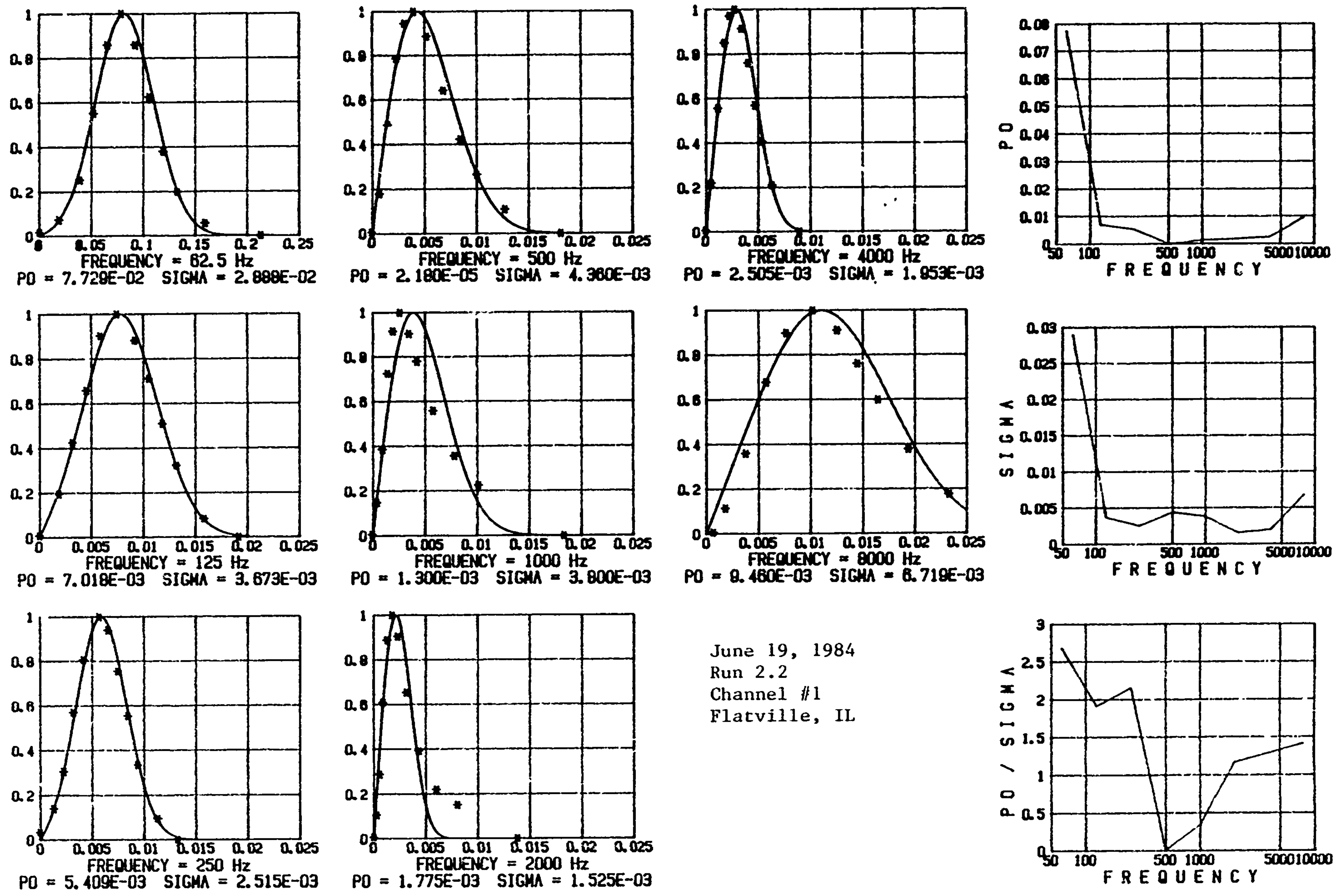
June 19, 1984
Run 1.2
Channel #3
Flatville, IL



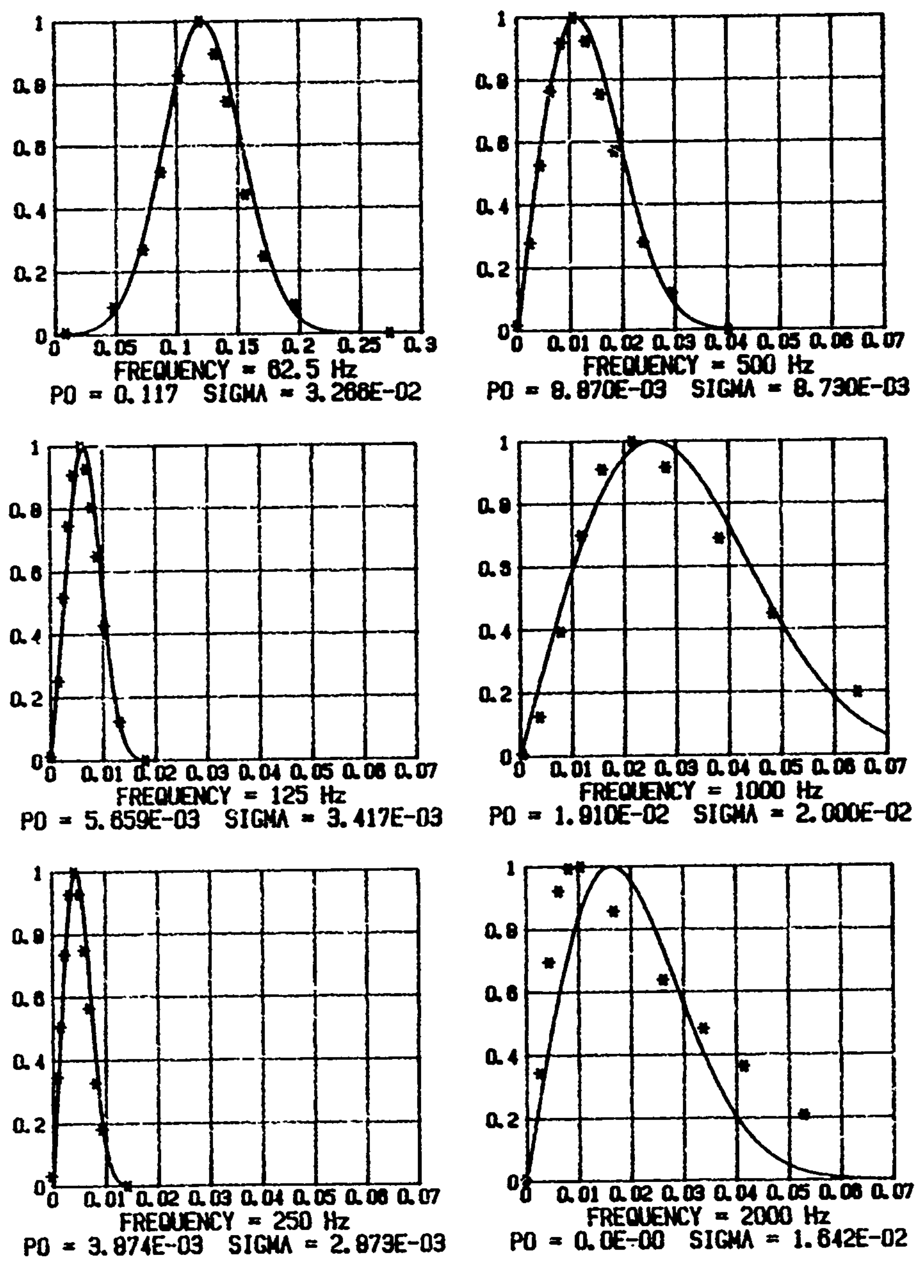
June 19, 1984
Run 1.2
Channel #4
Flatville, IL

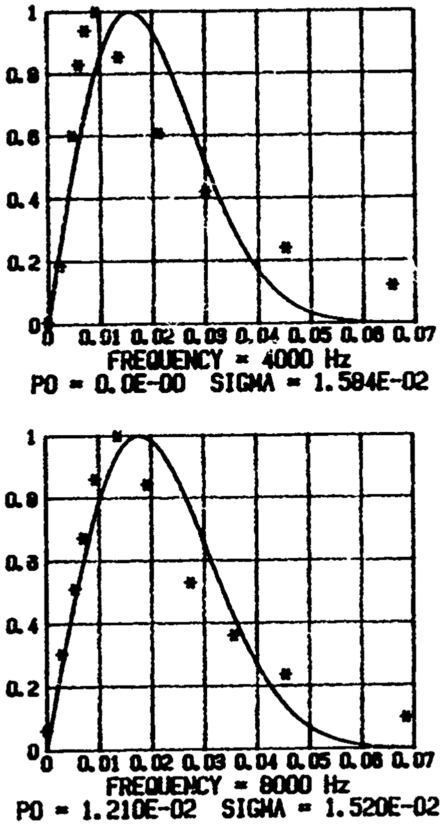


June 19, 1984
Run 1.2
Channel #5
Flatville, IL

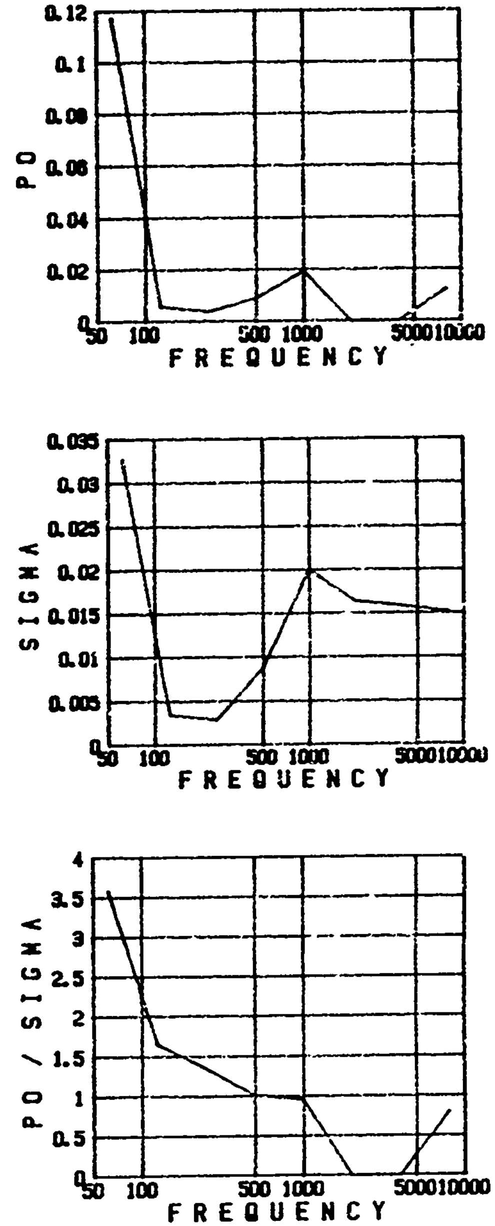


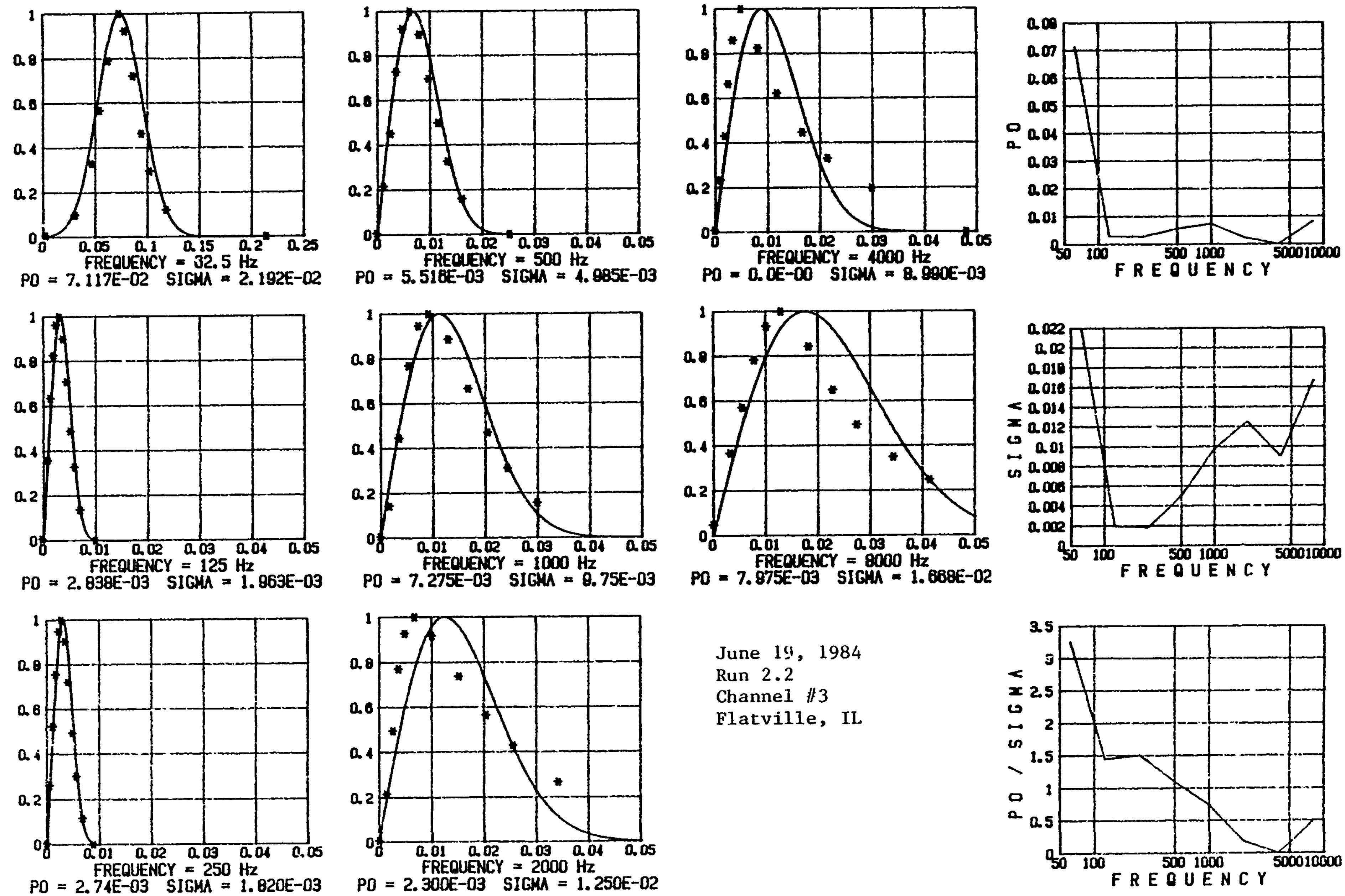
June 19, 1984
Run 2.2
Channel #1
Flatville, IL



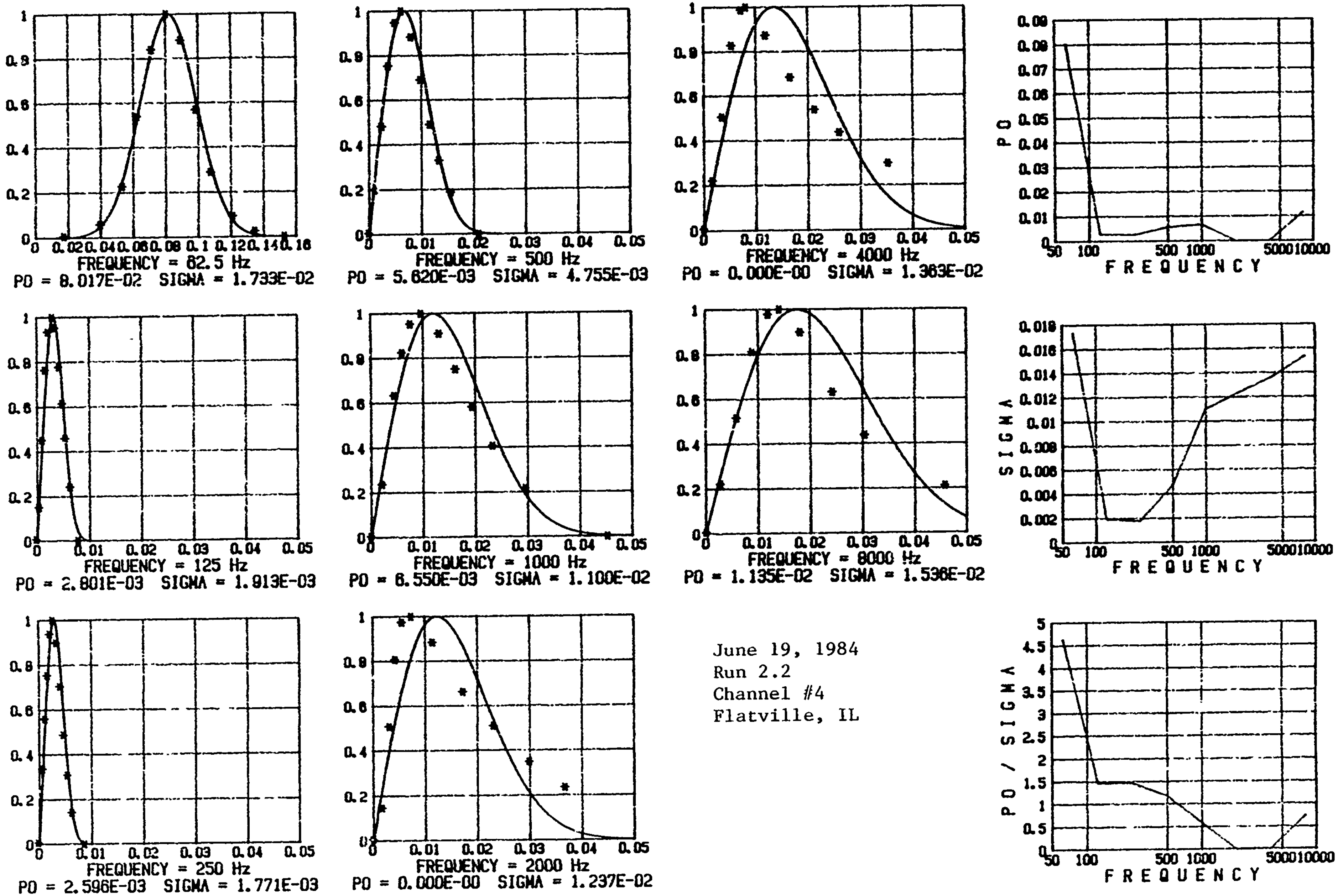


June 19, 1984
Run 2.2
Channel #2
Flatville, IL

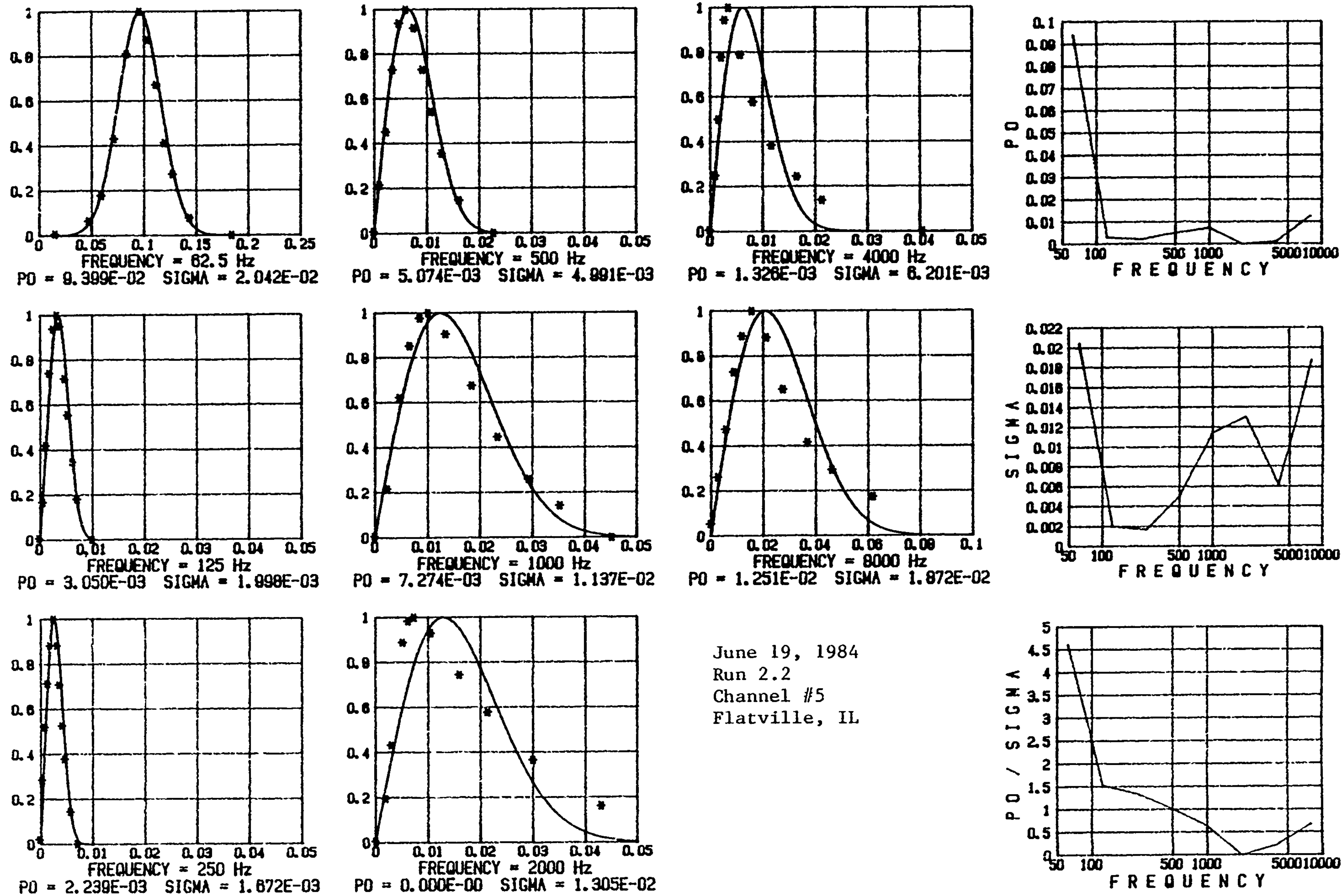




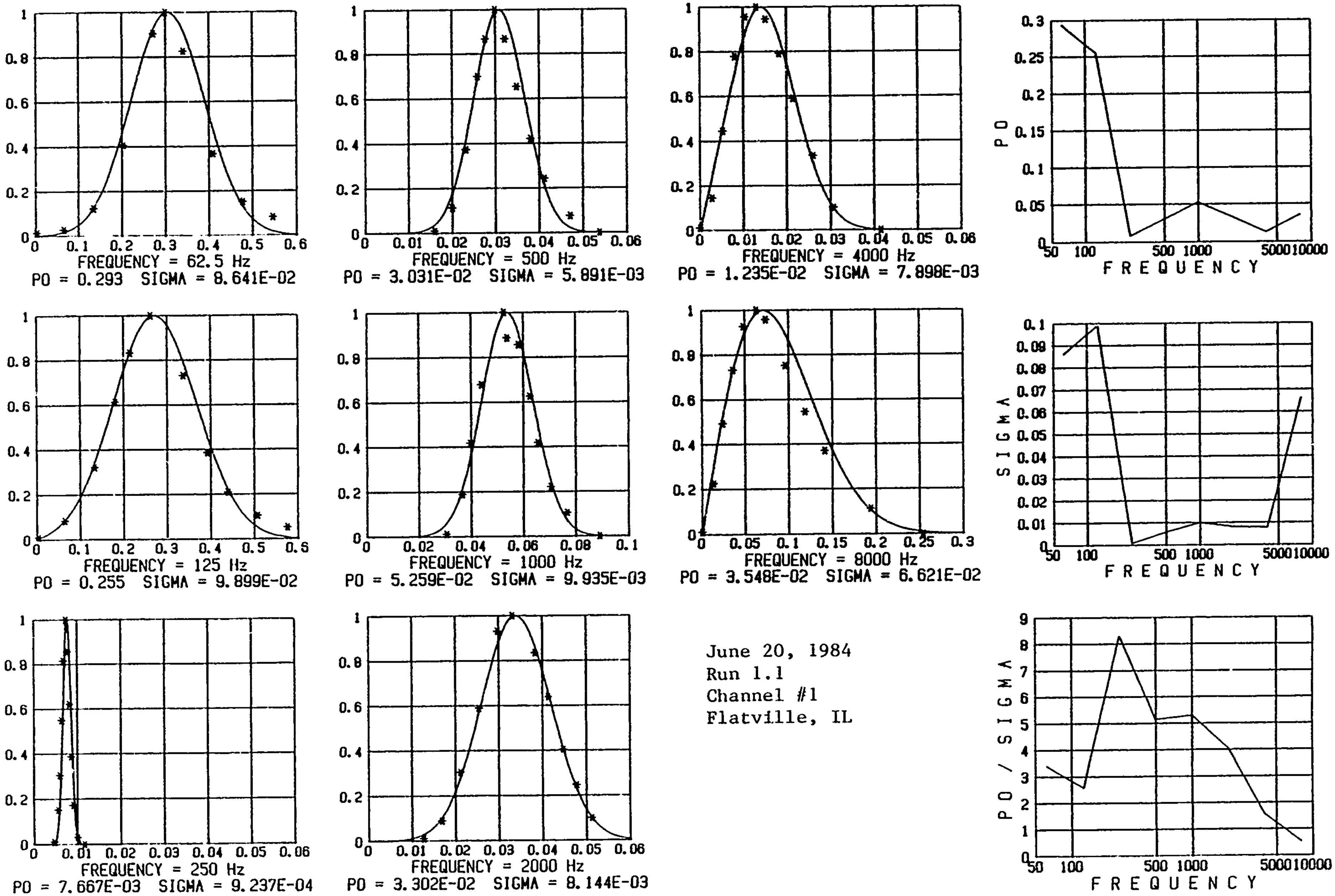
June 19, 1984
Run 2.2
Channel #3
Flatville, IL



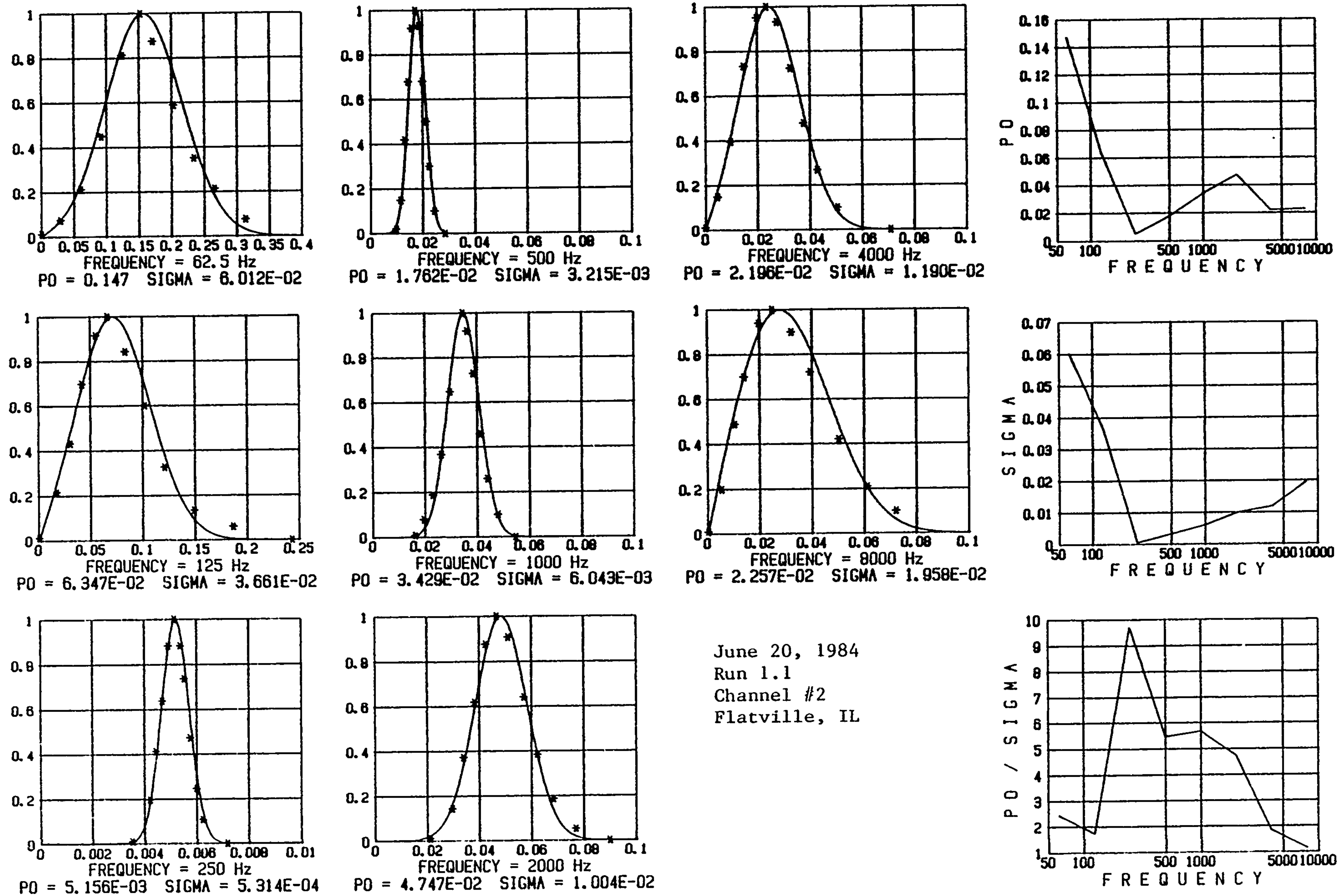
June 19, 1984
Run 2.2
Channel #4
Flatville, IL



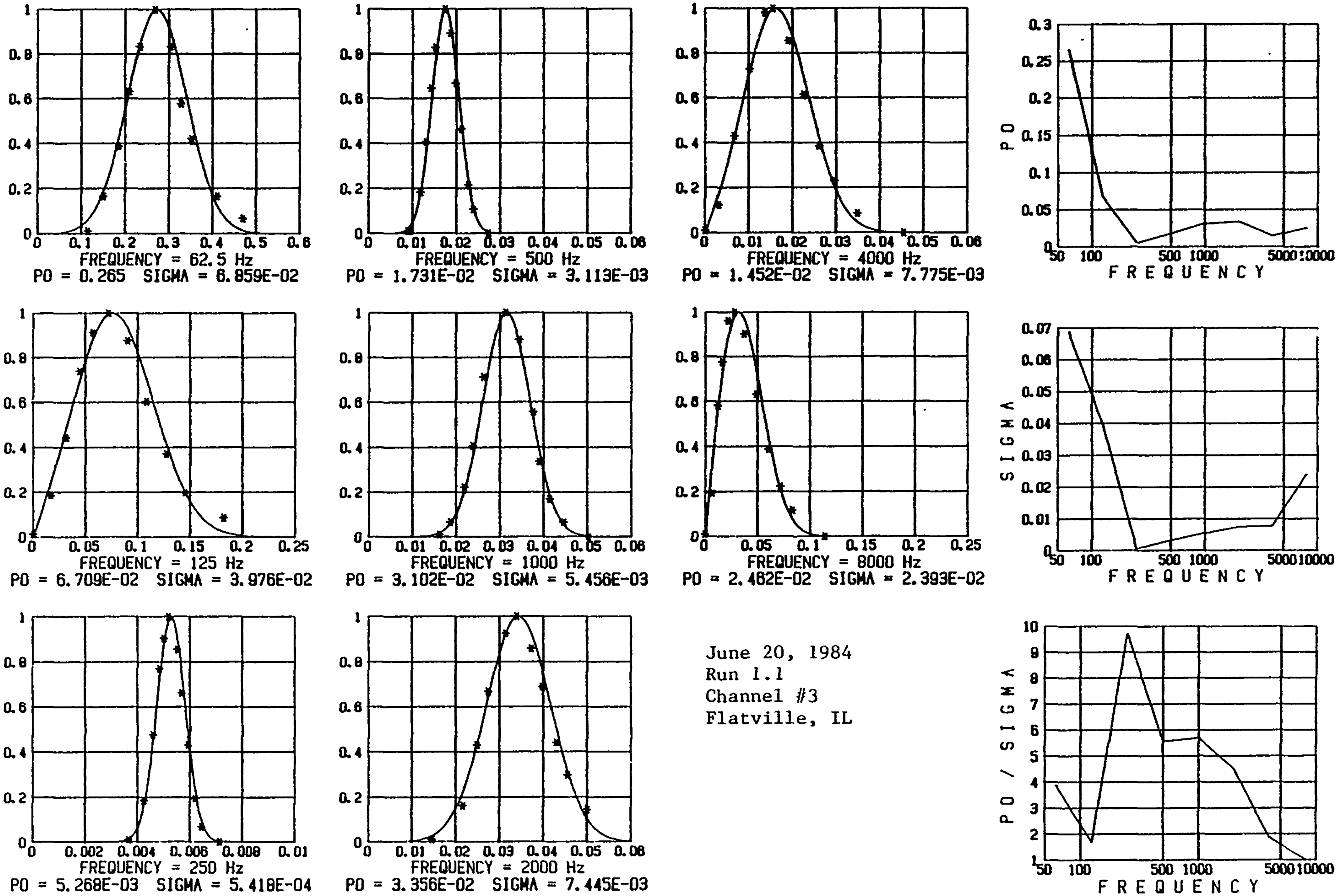
June 19, 1984
Run 2.2
Channel #5
Flatville, IL

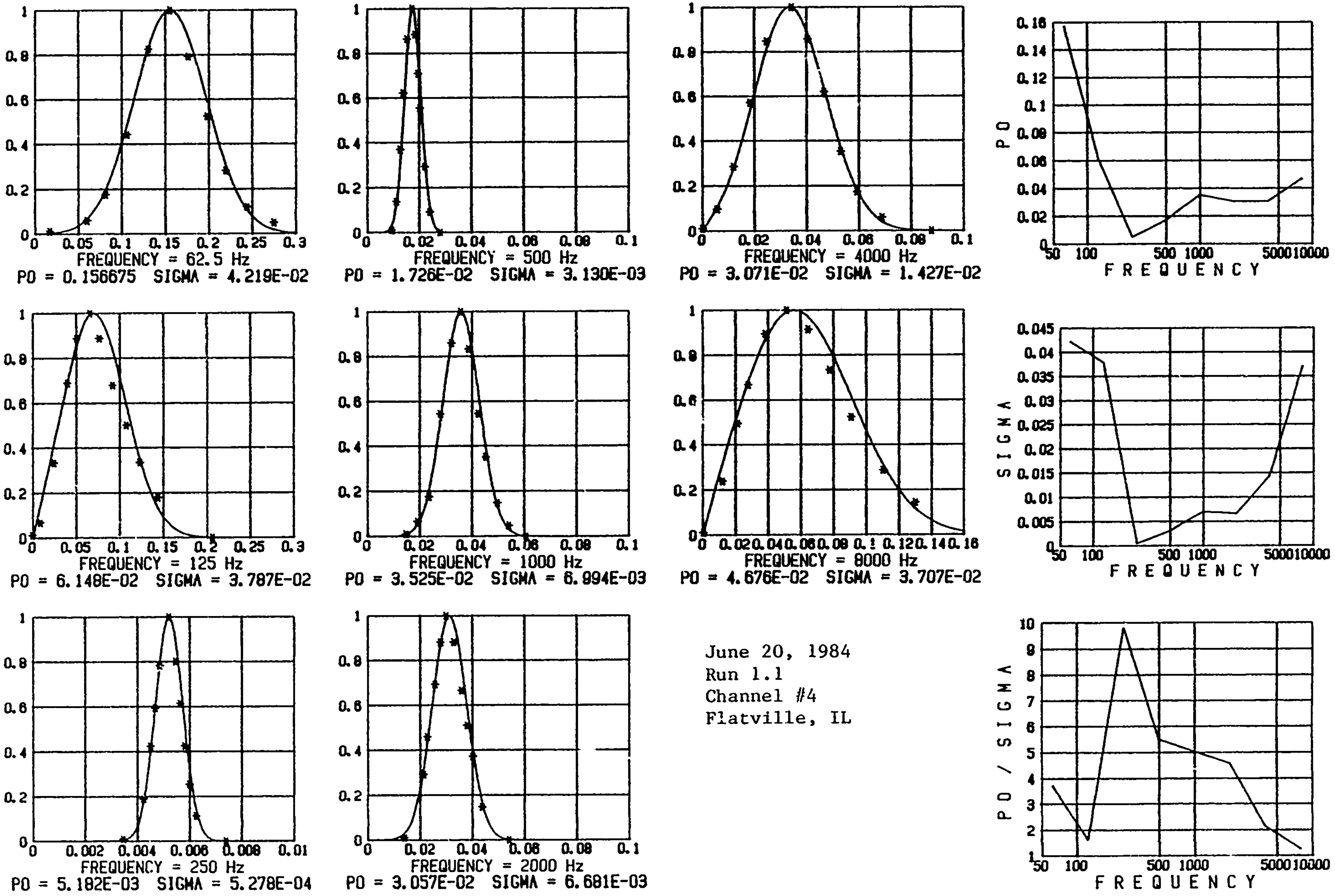


June 20, 1984
Run 1.1
Channel #1
Flatville, IL

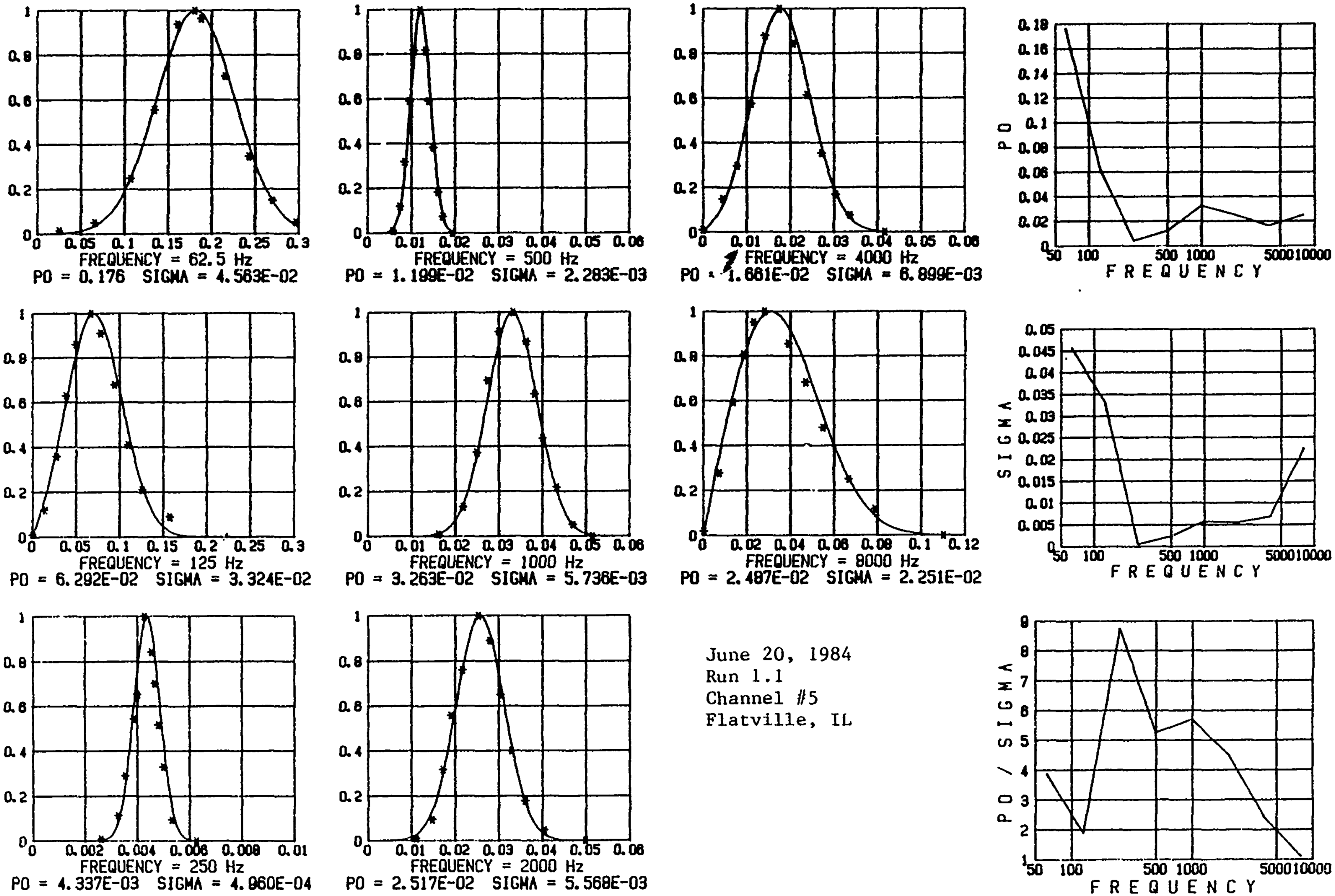


FREQUENCY = 62.5 Hz
PO = 0.147 SIGMA = 6.012E-02
FREQUENCY = 500 Hz
PO = 1.762E-02 SIGMA = 3.215E-03
FREQUENCY = 4000 Hz
PO = 2.196E-02 SIGMA = 1.190E-02
PO
FREQUENCY
FREQUENCY = 125 Hz
PO = 6.347E-02 SIGMA = 3.661E-02
FREQUENCY = 1000 Hz
PO = 3.429E-02 SIGMA = 6.043E-03
FREQUENCY = 8000 Hz
PO = 2.257E-02 SIGMA = 1.958E-02
SIGMA
FREQUENCY
FREQUENCY = 250 Hz
PO = 5.156E-03 SIGMA = 5.314E-04
FREQUENCY = 2000 Hz
PO = 4.747E-02 SIGMA = 1.004E-02
June 20, 1984
Run 1.1
Channel #2
Flatville, IL
PO / SIGMA
FREQUENCY

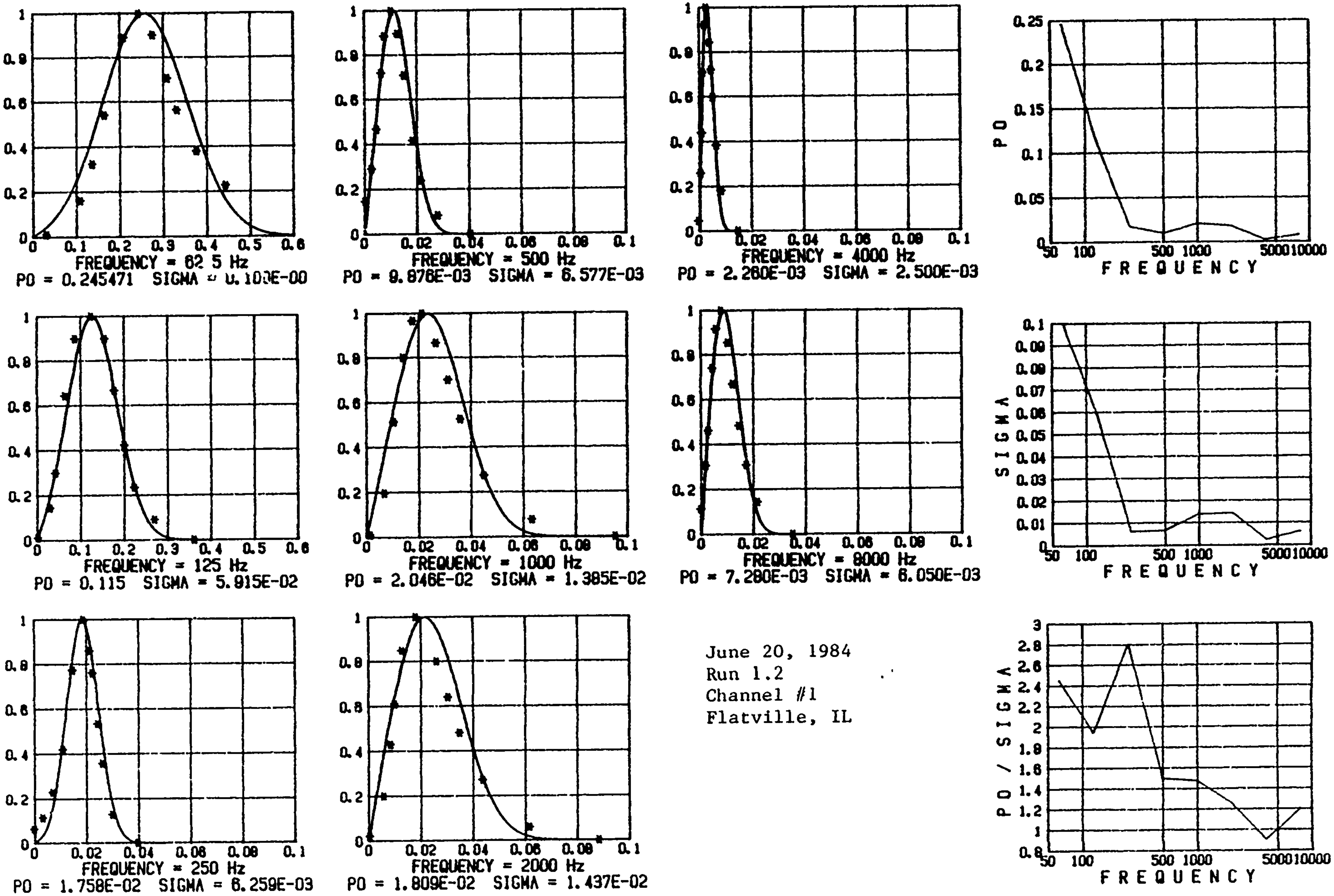

FREQUENCY = 62.5 Hz
PO = 0.265 SIGMA = 6.859E-02
FREQUENCY = 500 Hz
PO = 1.731E-02 SIGMA = 3.113E-03
FREQUENCY = 4000 Hz
PO = 1.452E-02 SIGMA = 7.775E-03
FREQUENCY = 125 Hz
PO = 6.709E-02 SIGMA = 3.976E-02
FREQUENCY = 1000 Hz
PO = 3.102E-02 SIGMA = 5.456E-03
FREQUENCY = 8000 Hz
PO = 2.462E-02 SIGMA = 2.393E-02
FREQUENCY = 250 Hz
PO = 5.268E-03 SIGMA = 5.418E-04
FREQUENCY = 2000 Hz
PO = 3.356E-02 SIGMA = 7.445E-03
PO
SIGMA
PO / SIGMA
FREQUENCY
June 20, 1984
Run 1.1
Channel #3
Flatville, IL

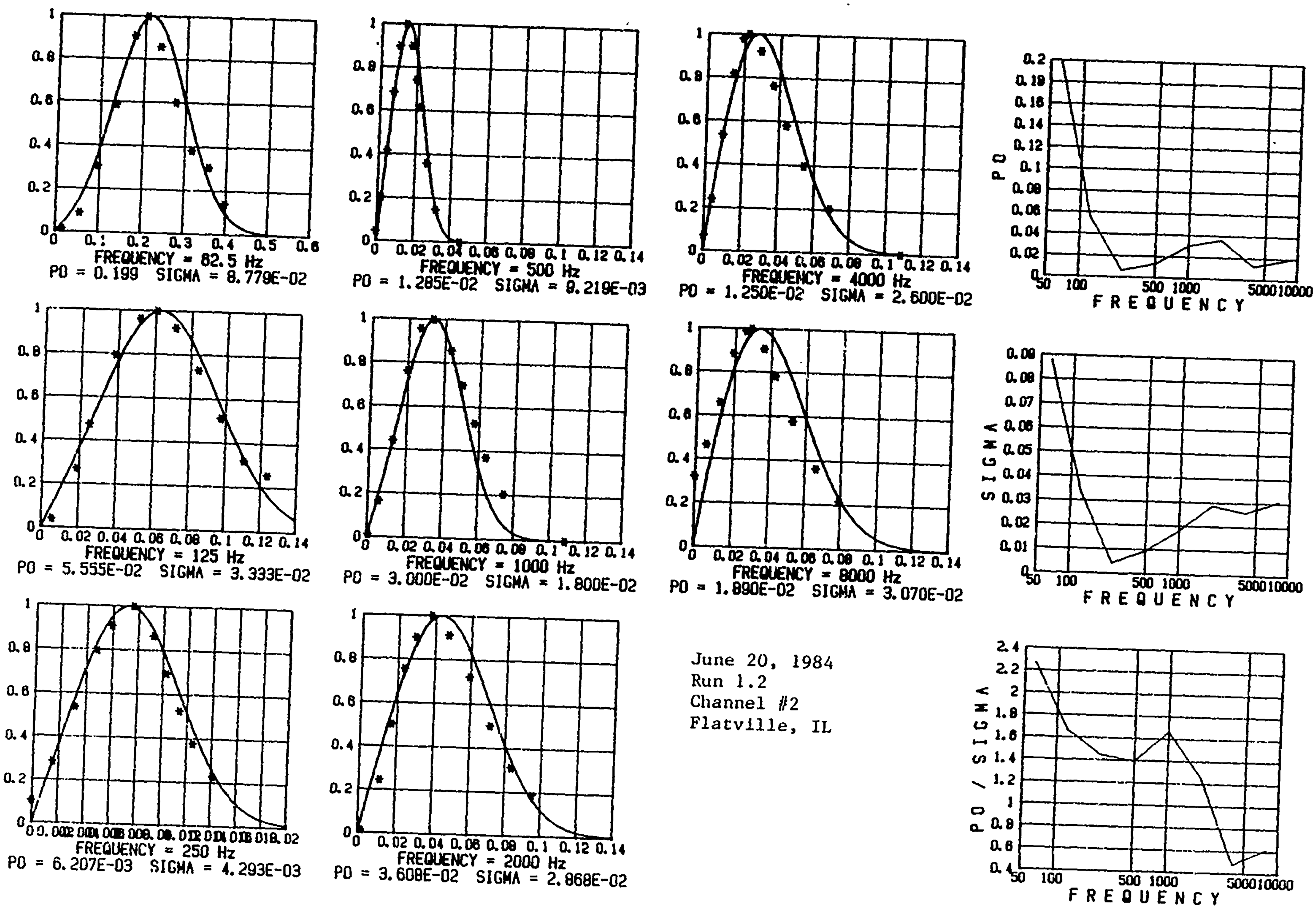


June 20, 1984
Run 1.1
Channel #4
Flatville, IL

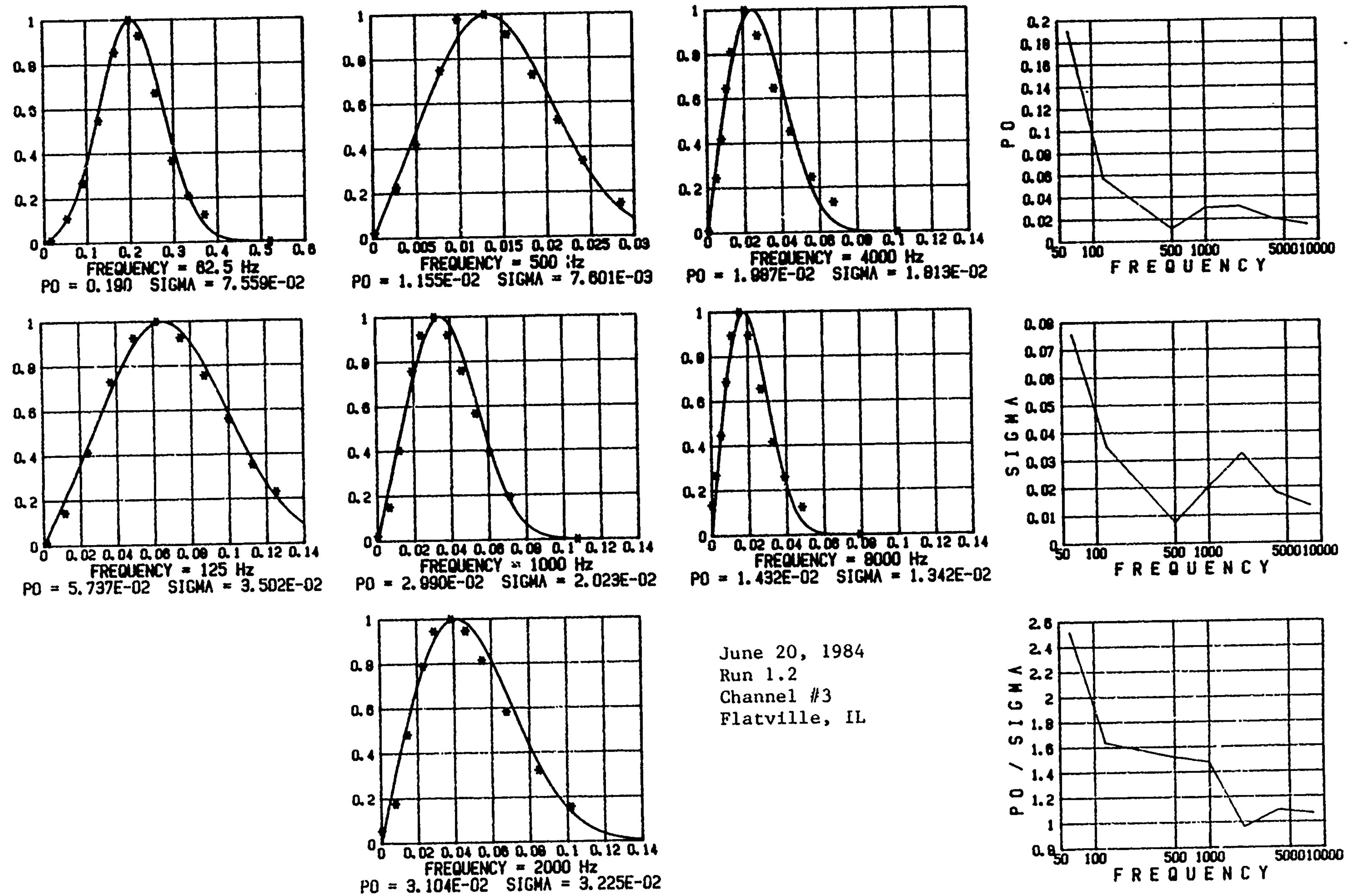


June 20, 1984
Run 1.1
Channel #5
Flatville, IL

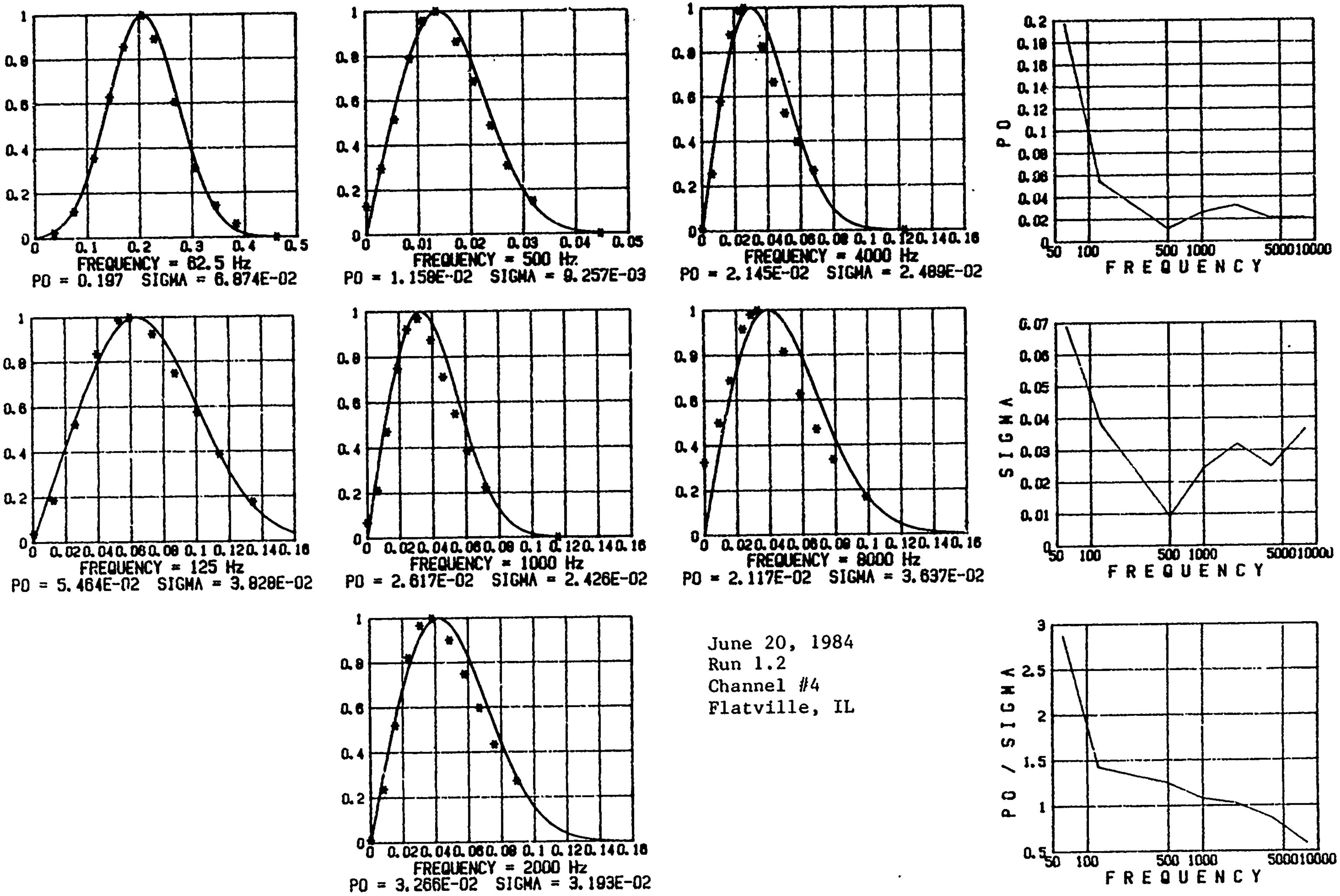

FREQUENCY = 62 5 Hz
PO = 0.245471 SIGMA = 0.10?E-00
FREQUENCY = 500 Hz
PO = 9.876E-03 SIGMA = 6.577E-03
FREQUENCY = 4000 Hz
PO = 2.260E-03 SIGMA = 2.500E-03
FREQUENCY = 125 Hz
PO = 0.115 SIGMA = 5.915E-02
FREQUENCY = 1000 Hz
PO = 2.046E-02 SIGMA = 1.385E-02
FREQUENCY = 8000 Hz
PO = 7.280E-03 SIGMA = 6.050E-03
FREQUENCY = 250 Hz
PO = 1.758E-02 SIGMA = 6.259E-03
FREQUENCY = 2000 Hz
PO = 1.809E-02 SIGMA = 1.437E-02
PO
SIGMA
PO / SIGMA
FREQUENCY
June 20, 1984
Run 1.2
Channel #1
Flatville, IL



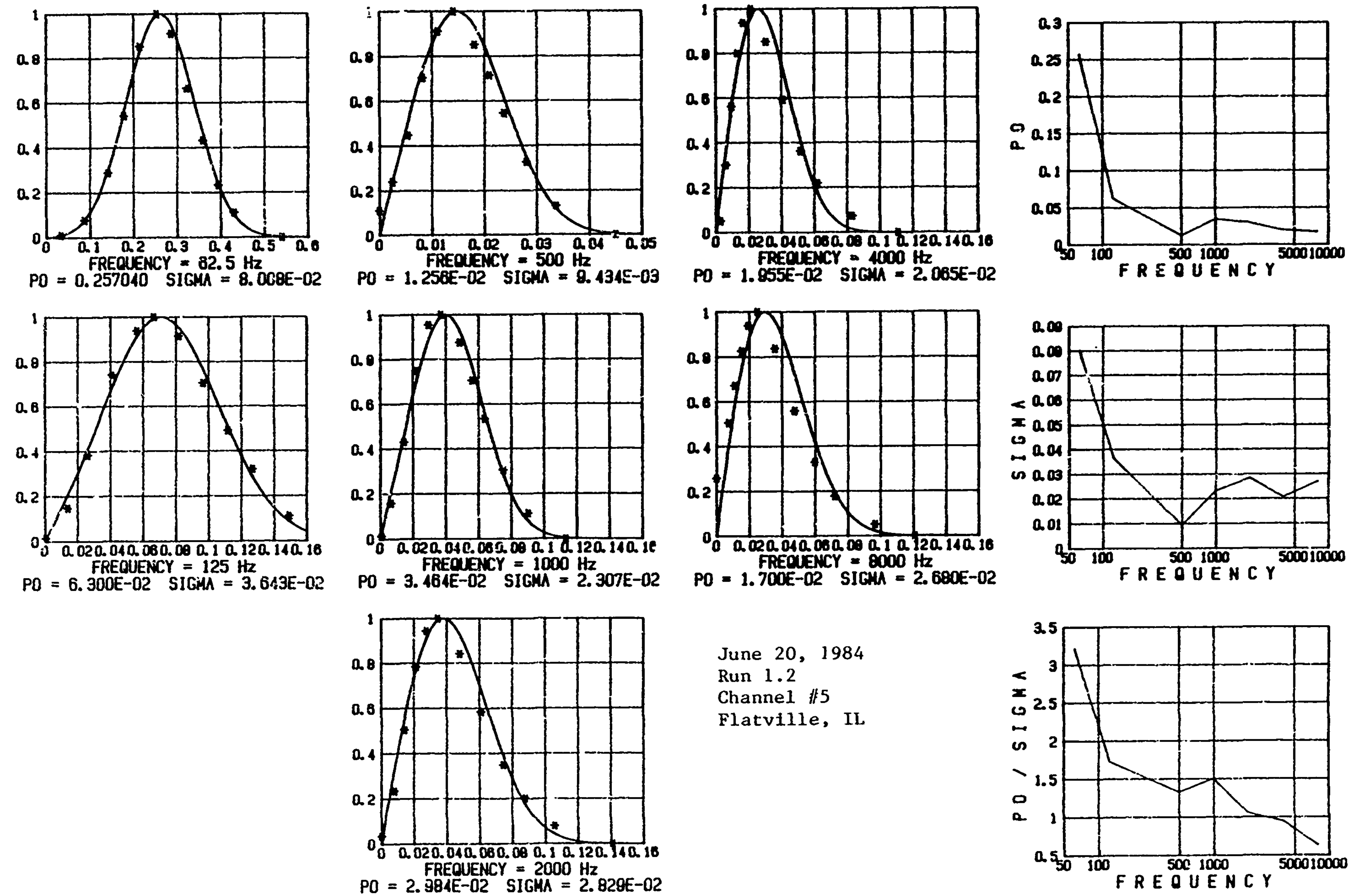
June 20, 1984
Run 1.2
Channel #2
Flatville, IL



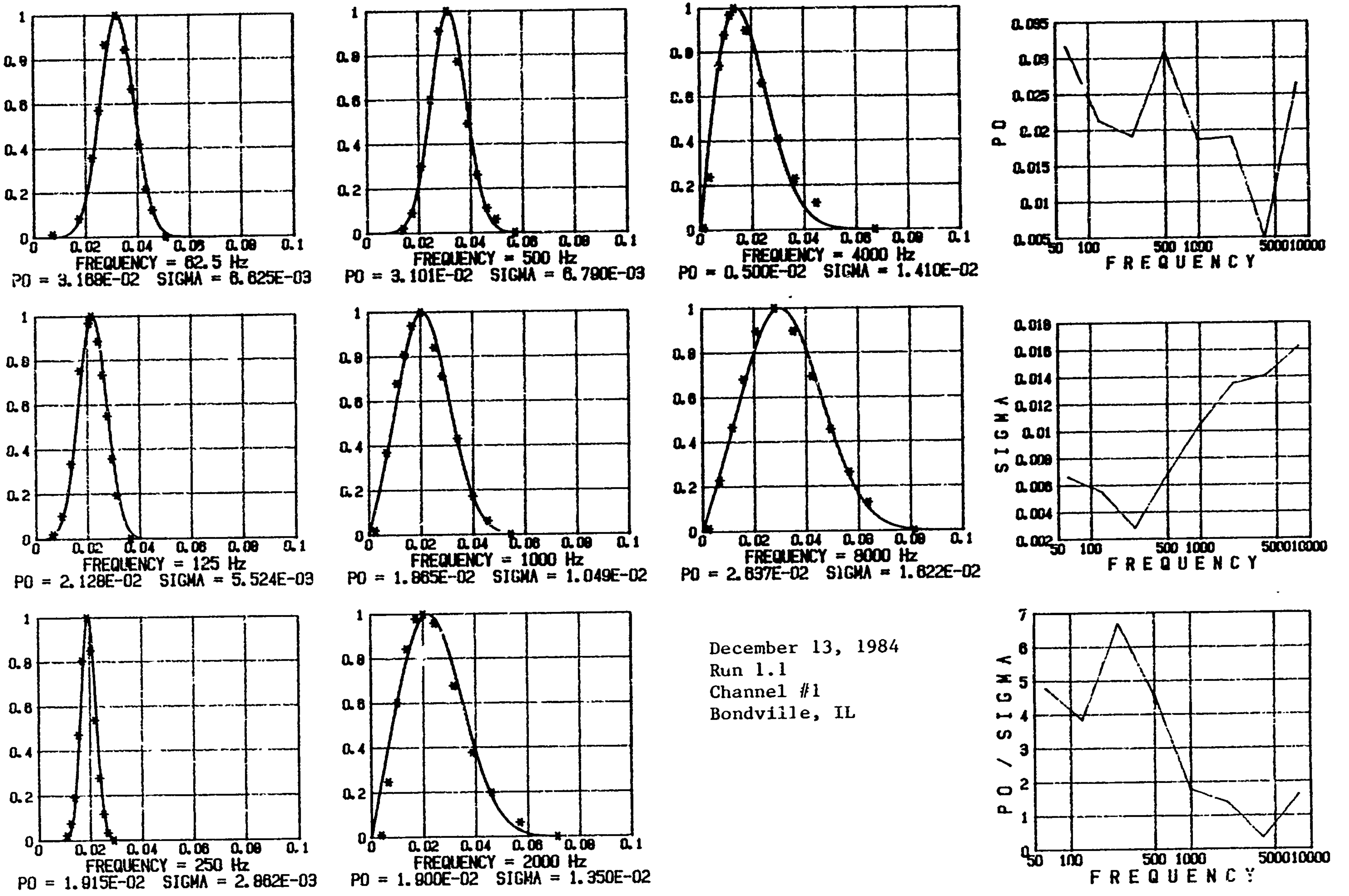
June 20, 1984
Run 1.2
Channel #3
Flatville, IL



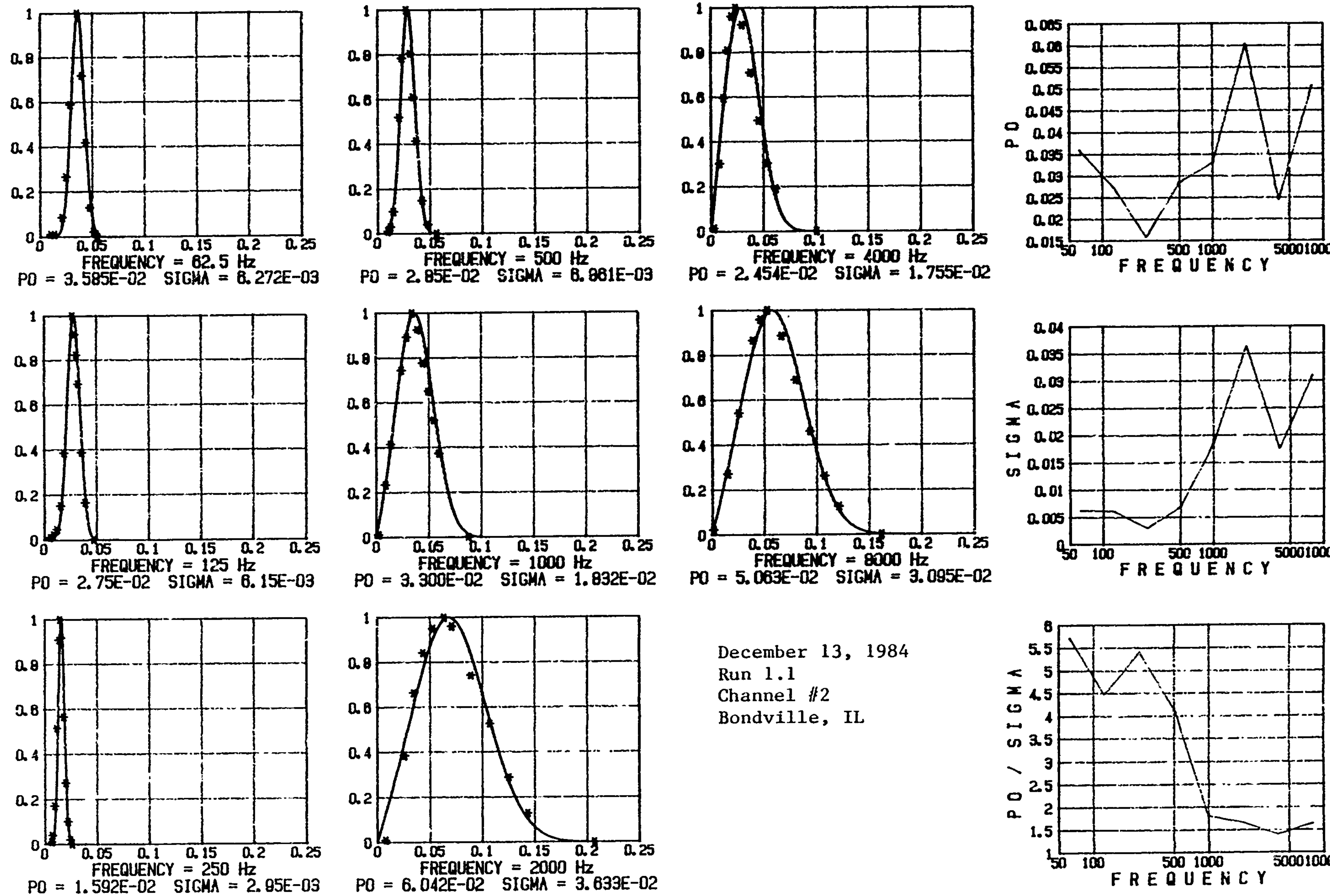
June 20, 1984
Run 1.2
Channel #4
Flatville, IL

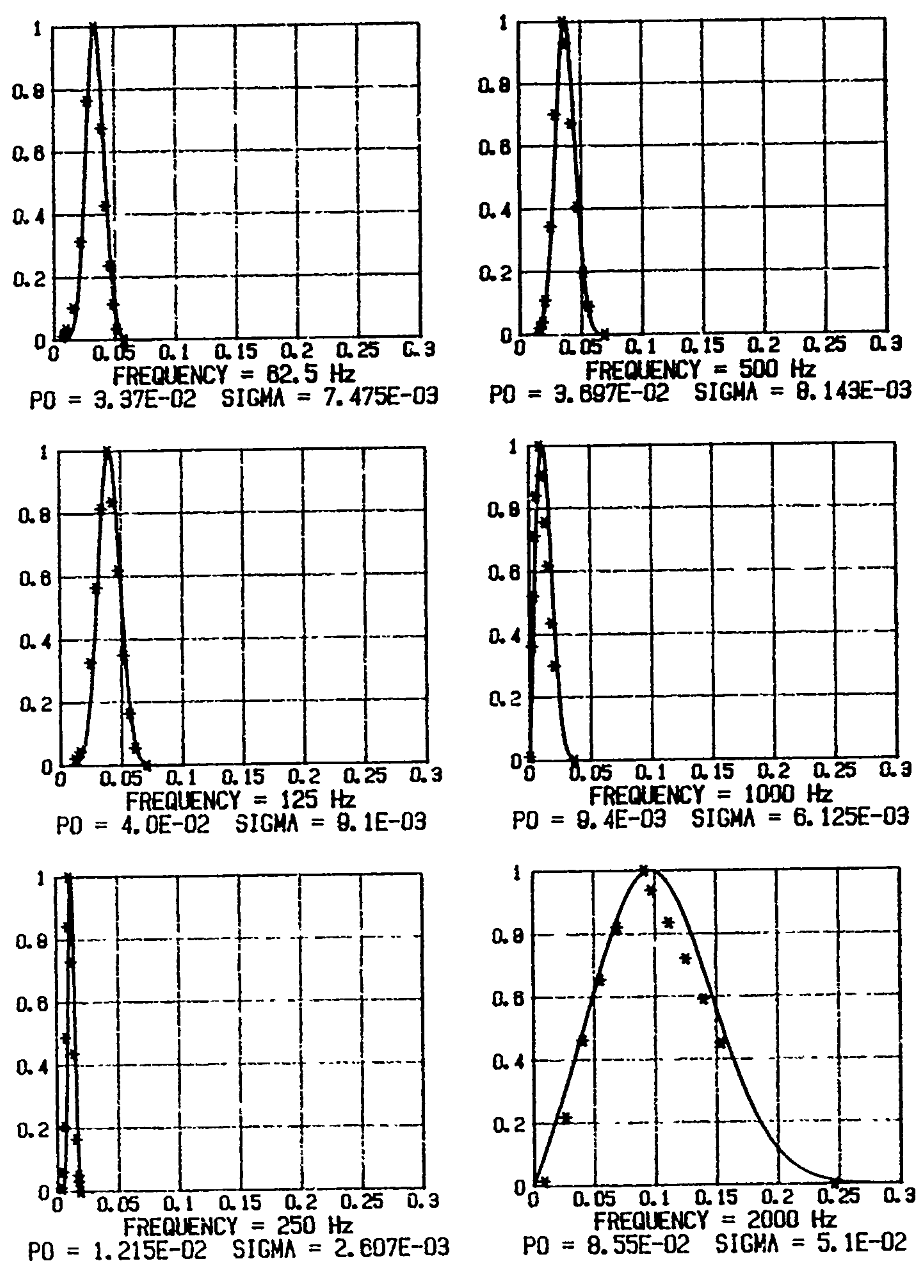


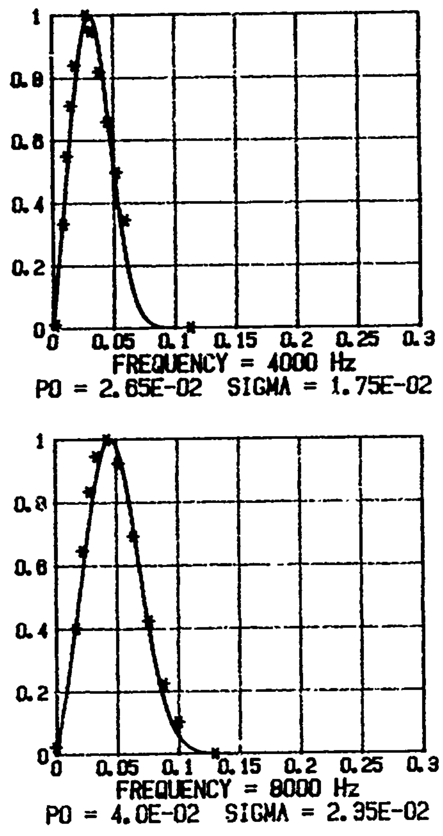
June 20, 1984
Run 1.2
Channel #5
Flatville, IL

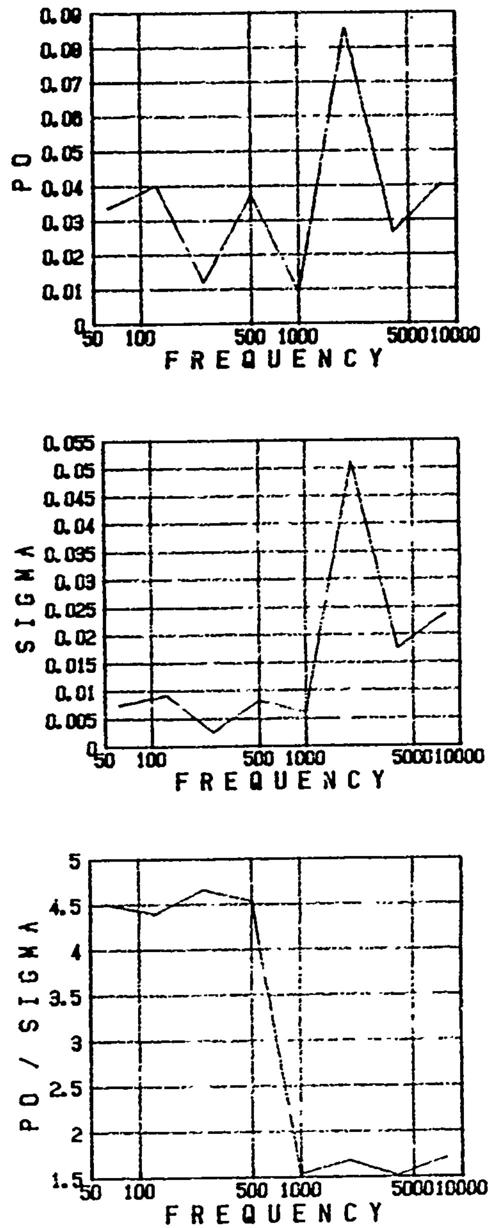


December 13, 1984
Run 1.1
Channel #1
Bondvilie, IL

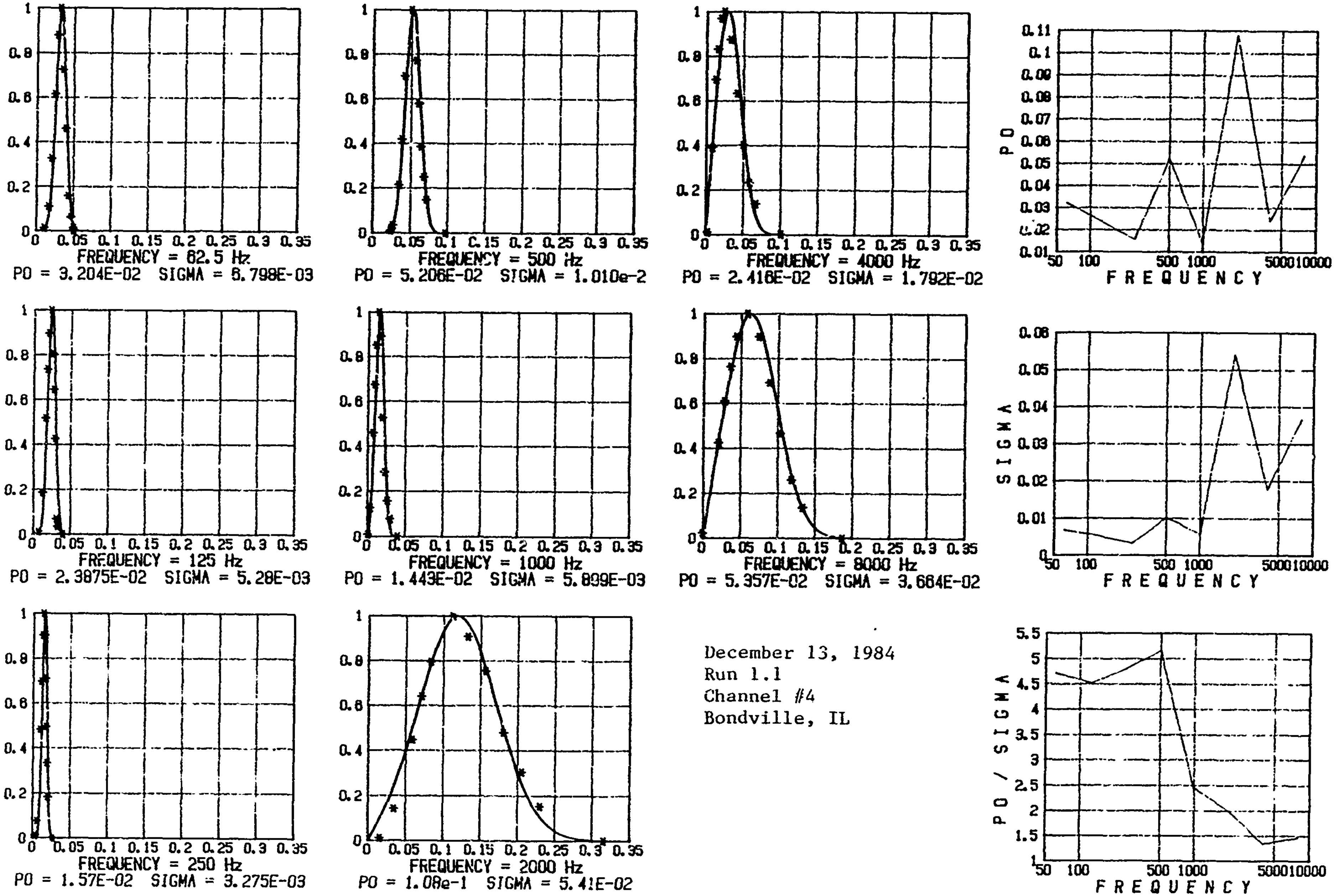

FREQUENCY = 62.5 Hz
PO = 3.585E-02 SIGMA = 6.272E-03
FREQUENCY = 500 Hz
PO = 2.85E-02 SIGMA = 6.881E-03
FREQUENCY = 4000 Hz
PO = 2.454E-02 SIGMA = 1.755E-02
P O
F R E Q U E N C Y
FREQUENCY = 125 Hz
PO = 2.75E-02 SIGMA = 6.15E-03
FREQUENCY = 1000 Hz
PO = 3.300E-02 SIGMA = 1.832E-02
FREQUENCY = 8000 Hz
PO = 5.063E-02 SIGMA = 3.095E-02
S I G M A
F R E Q U E N C Y
FREQUENCY = 250 Hz
PO = 1.592E-02 SIGMA = 2.85E-03
FREQUENCY = 2000 Hz
PO = 6.042E-02 SIGMA = 3.633E-02
December 13, 1984
Run 1.1
Channel #2
Bondville, IL
P O / S I G M A
F R E Q U E N C Y



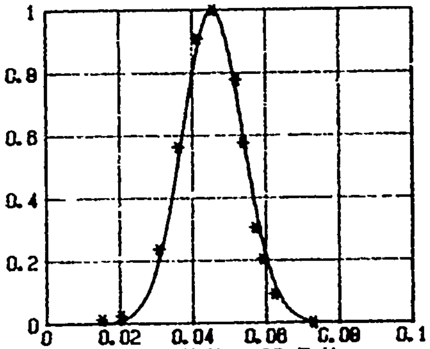




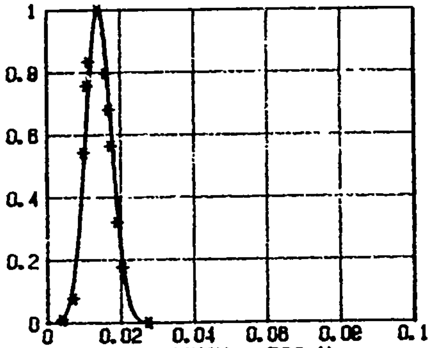
December 13, 1984
Run 1.1
Channel #3
Bondville, IL



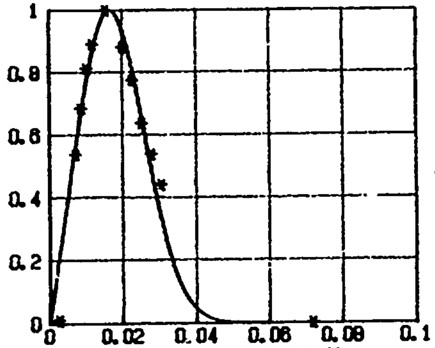
December 13, 1984
Run 1.1
Channel #4
Bondville, IL



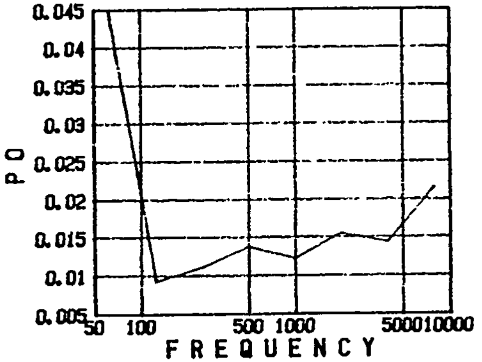
FREQUENCY = 62.5 Hz
PO = 4.495E-02   SIGMA = 8.288E-03

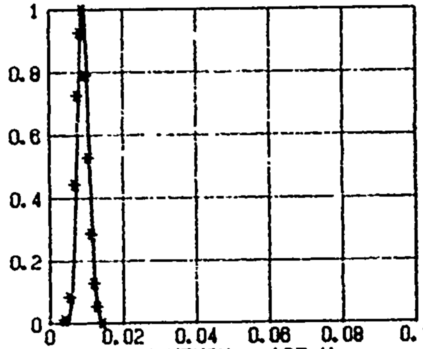


FREQUENCY = 500 Hz
PO = 1.375E-02   SIGMA = 3.55E-03

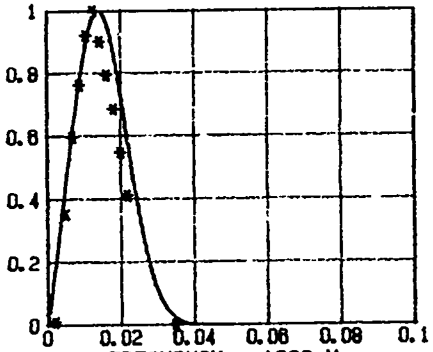


FREQUENCY = 4000 Hz
PO = 1.4375E-02   SIGMA = 9.725E-03

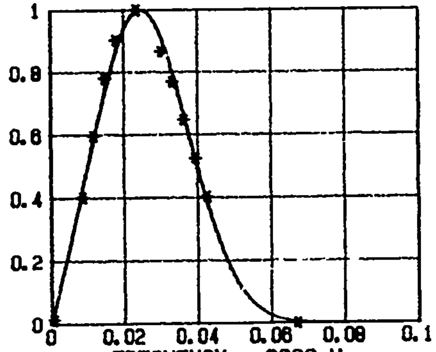




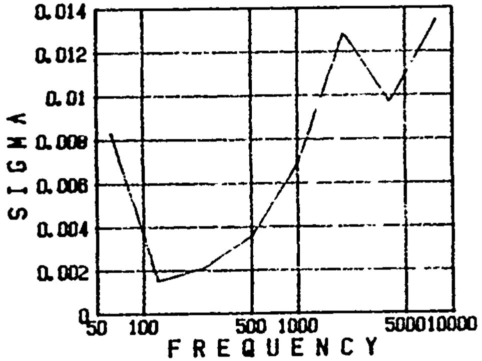
FREQUENCY = 125 Hz
PO = 9.255E-03   SIGMA = 1.525E-03

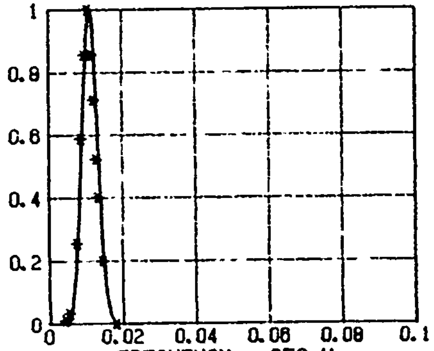


FREQUENCY = 1000 Hz
PO = 1.227E-02   SIGMA = 6.825E-03

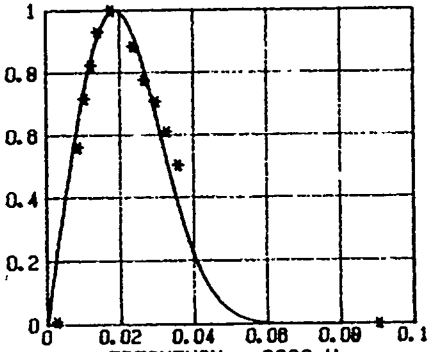


FREQUENCY = 8000 Hz
PO = 2.15E-02   SIGMA = 1.345E-02



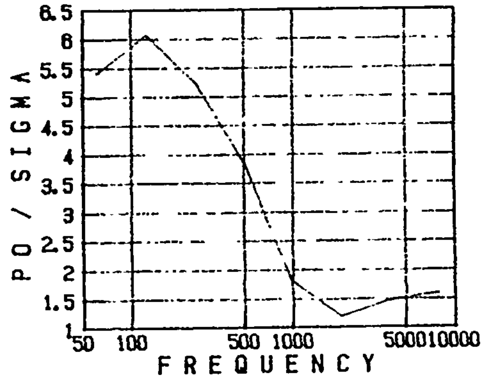


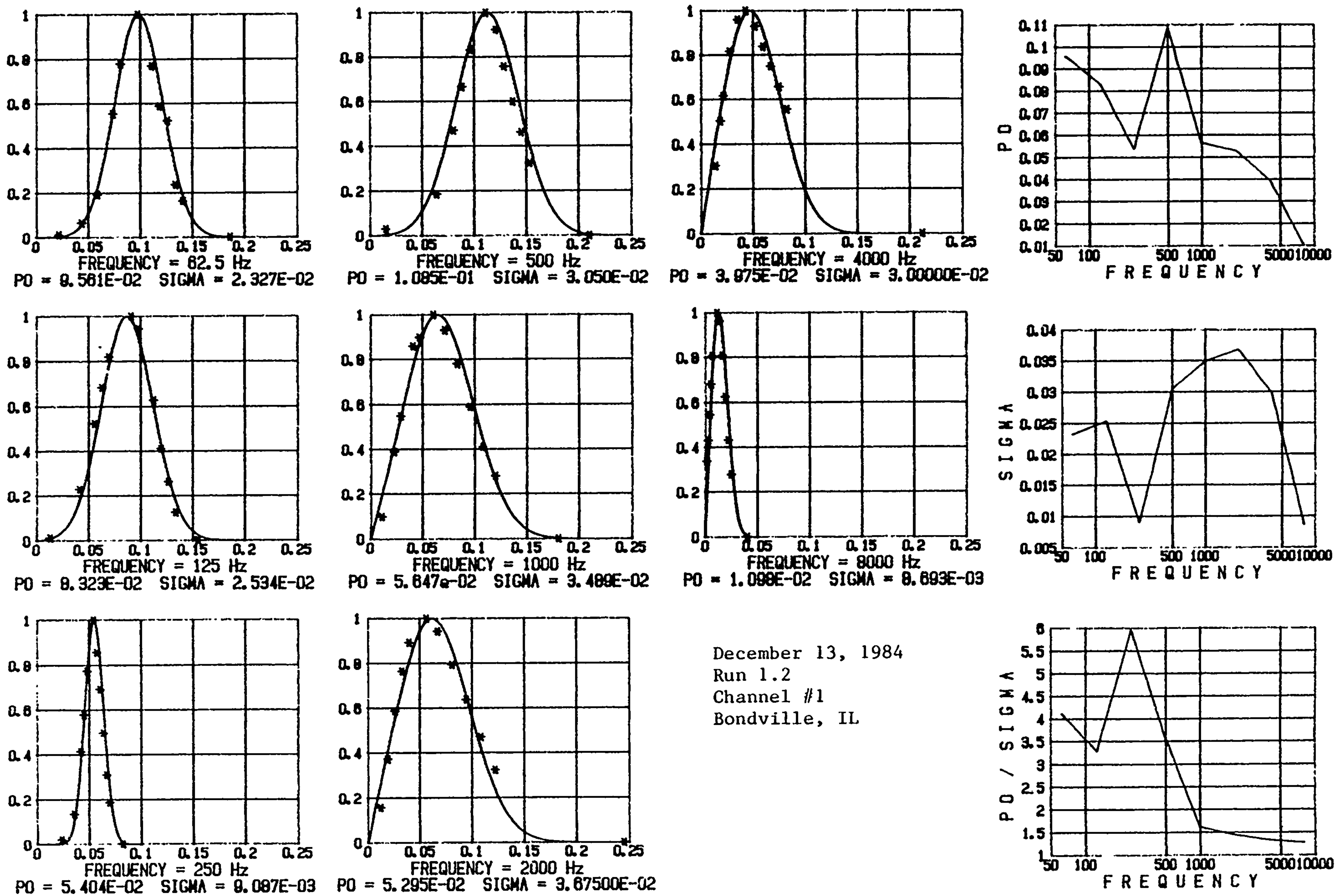
FREQUENCY = 250 Hz
PO = 1.112E-02   SIGMA = 2.116E-03



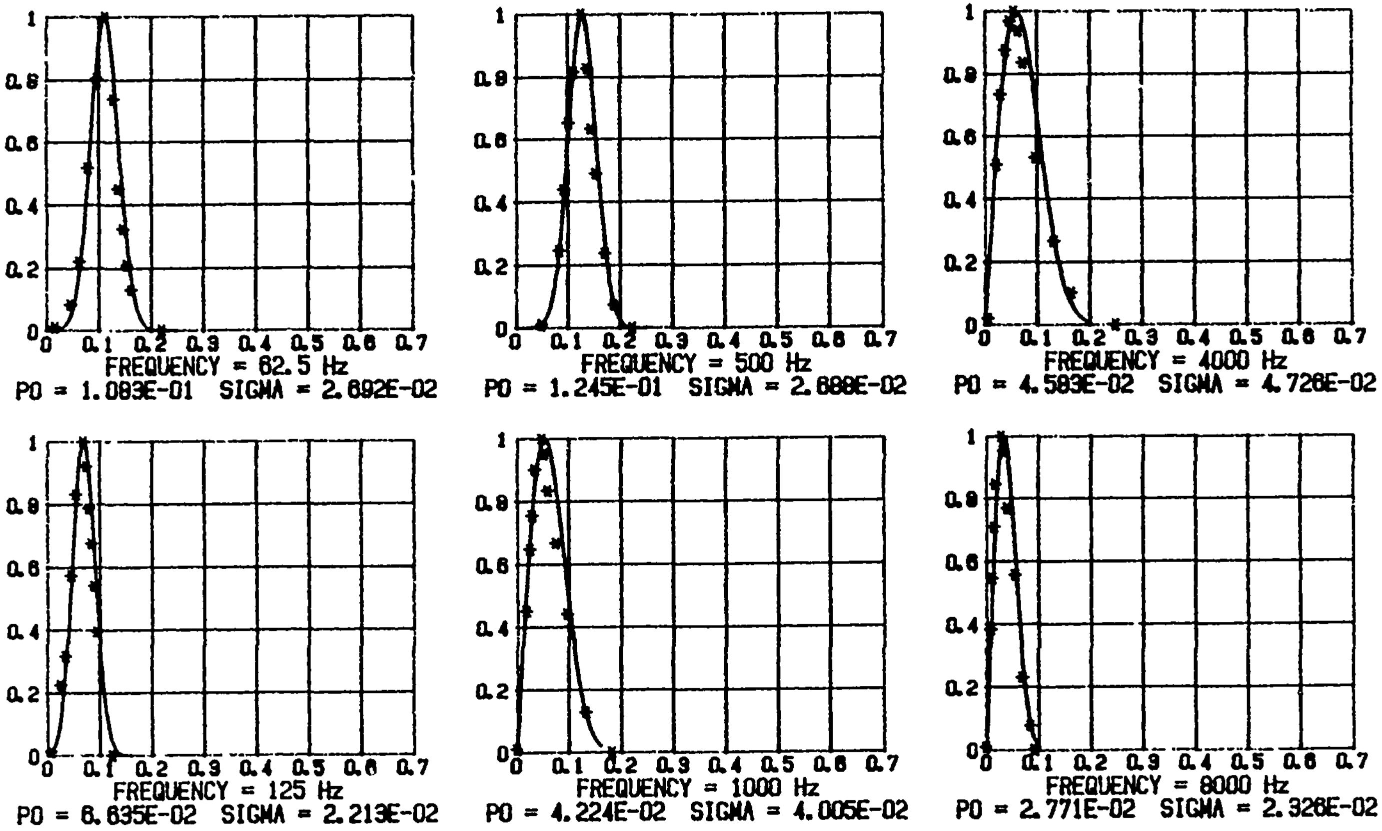
FREQUENCY = 2000 Hz
PO = 1.55E-02   SIGMA = 1.285E-02

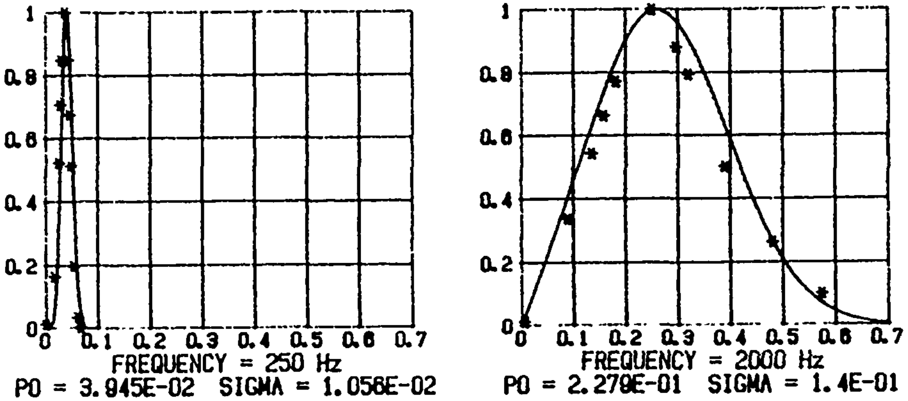
December 13, 1984
Run 1.1
Channel #5
Bondville, IL

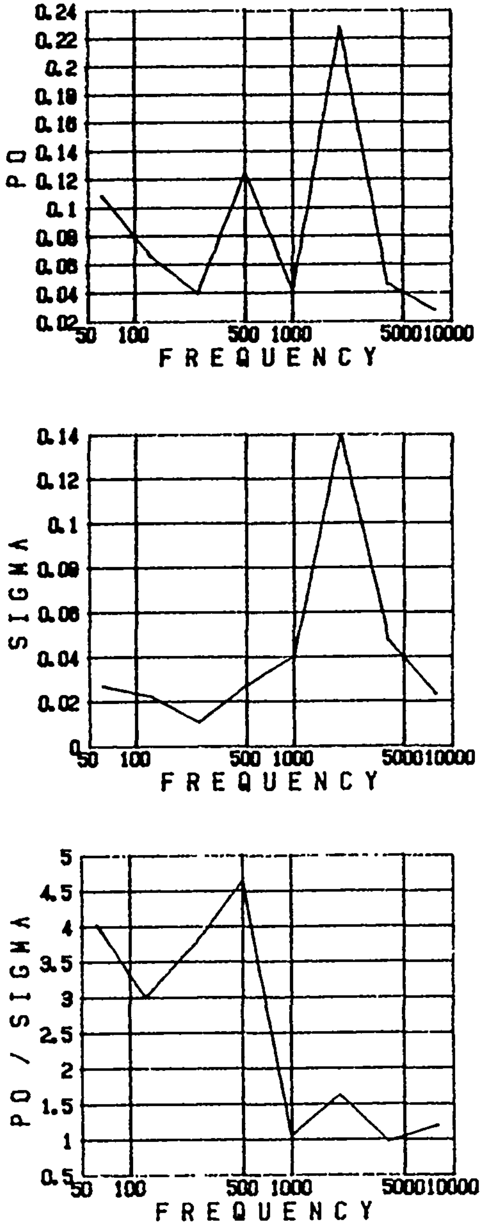




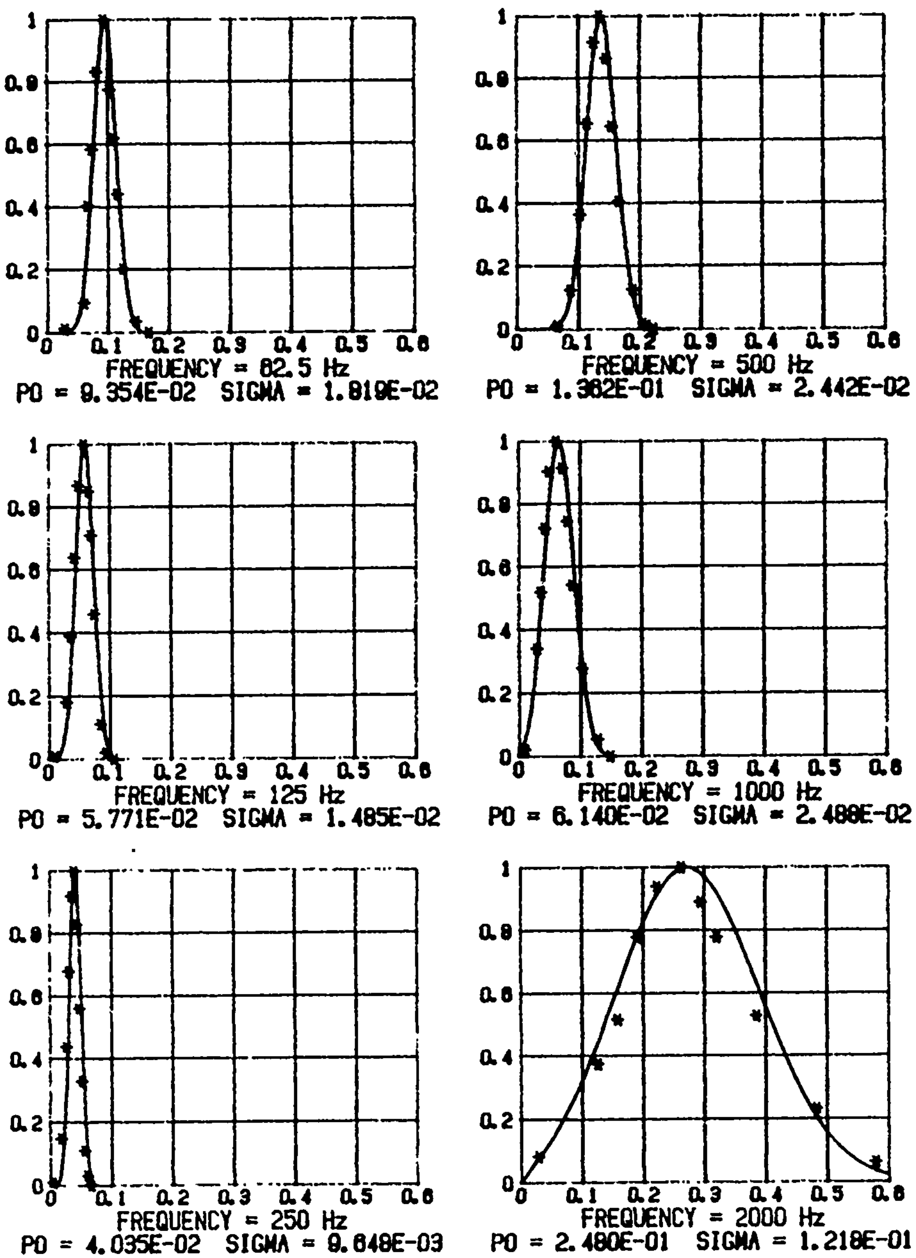
December 13, 1984
Run 1.2
Channel #1
Bondville, IL

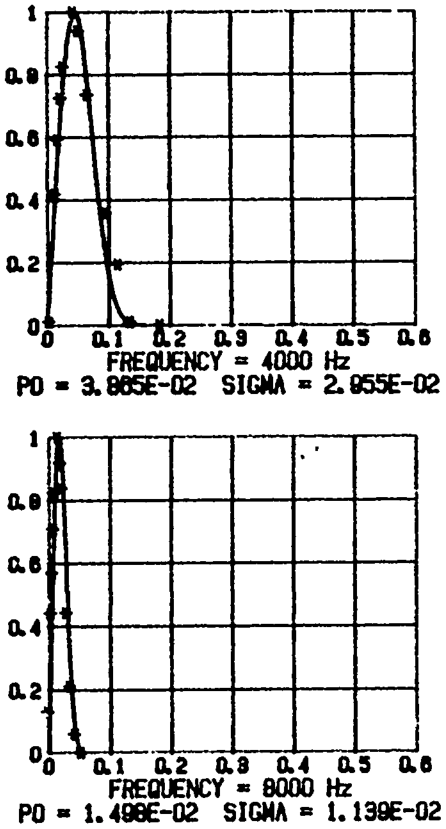


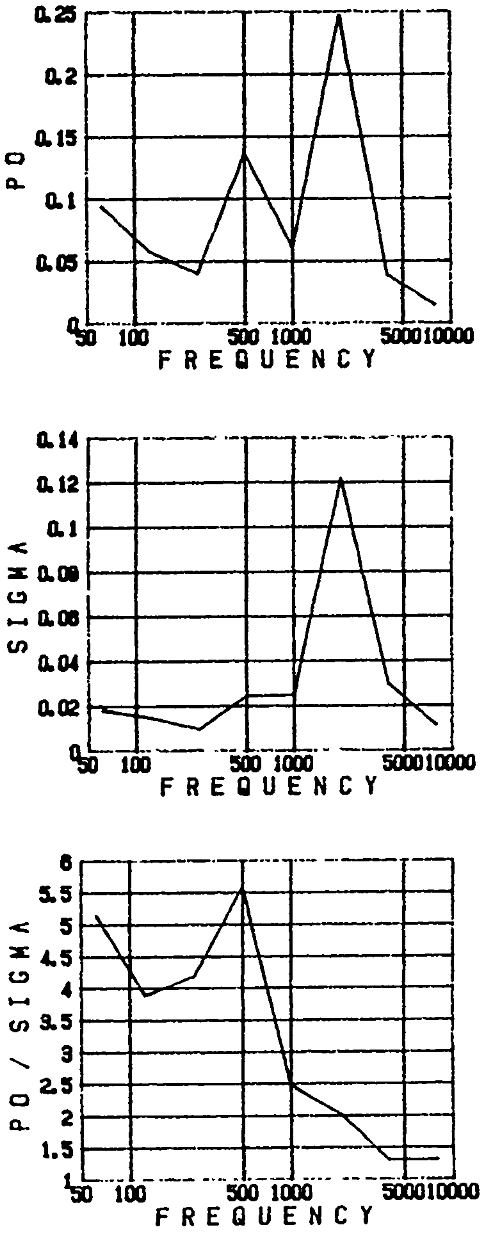




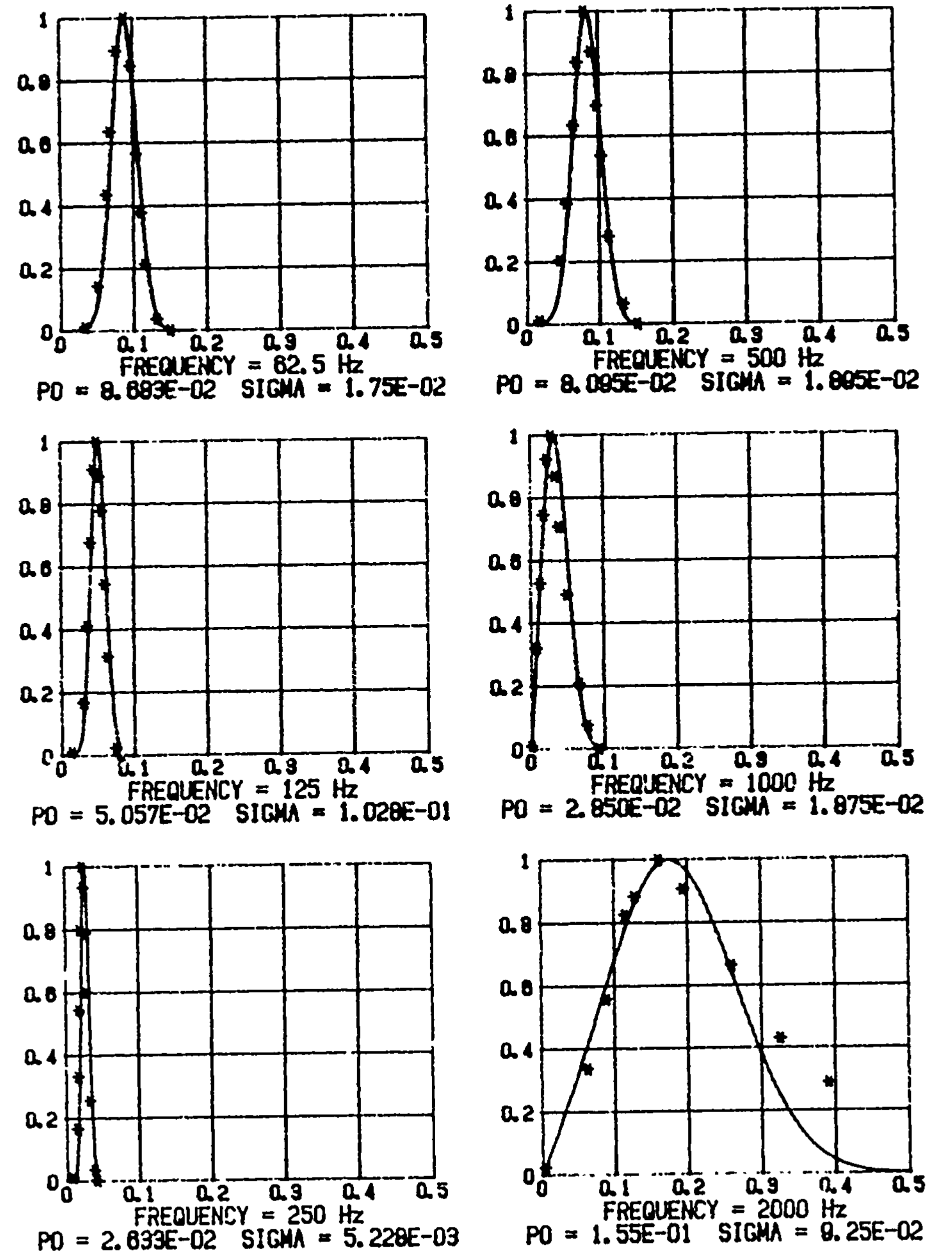
December 13, 1984
Run 1.2
Channel #3
Bondville, IL

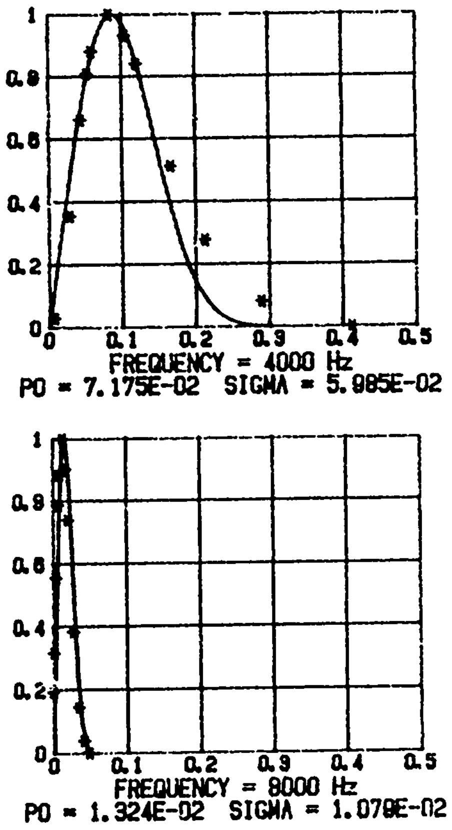




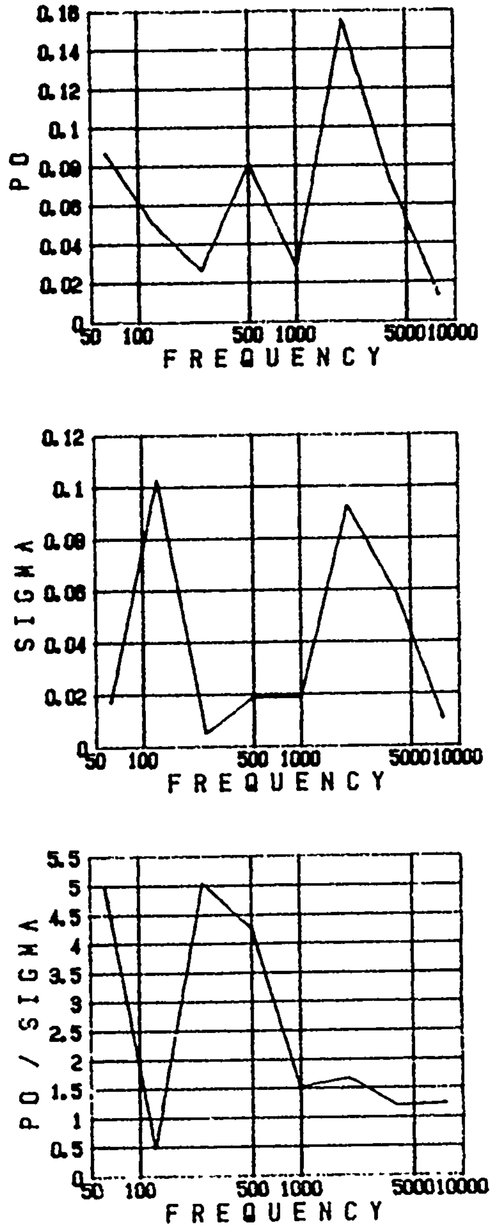


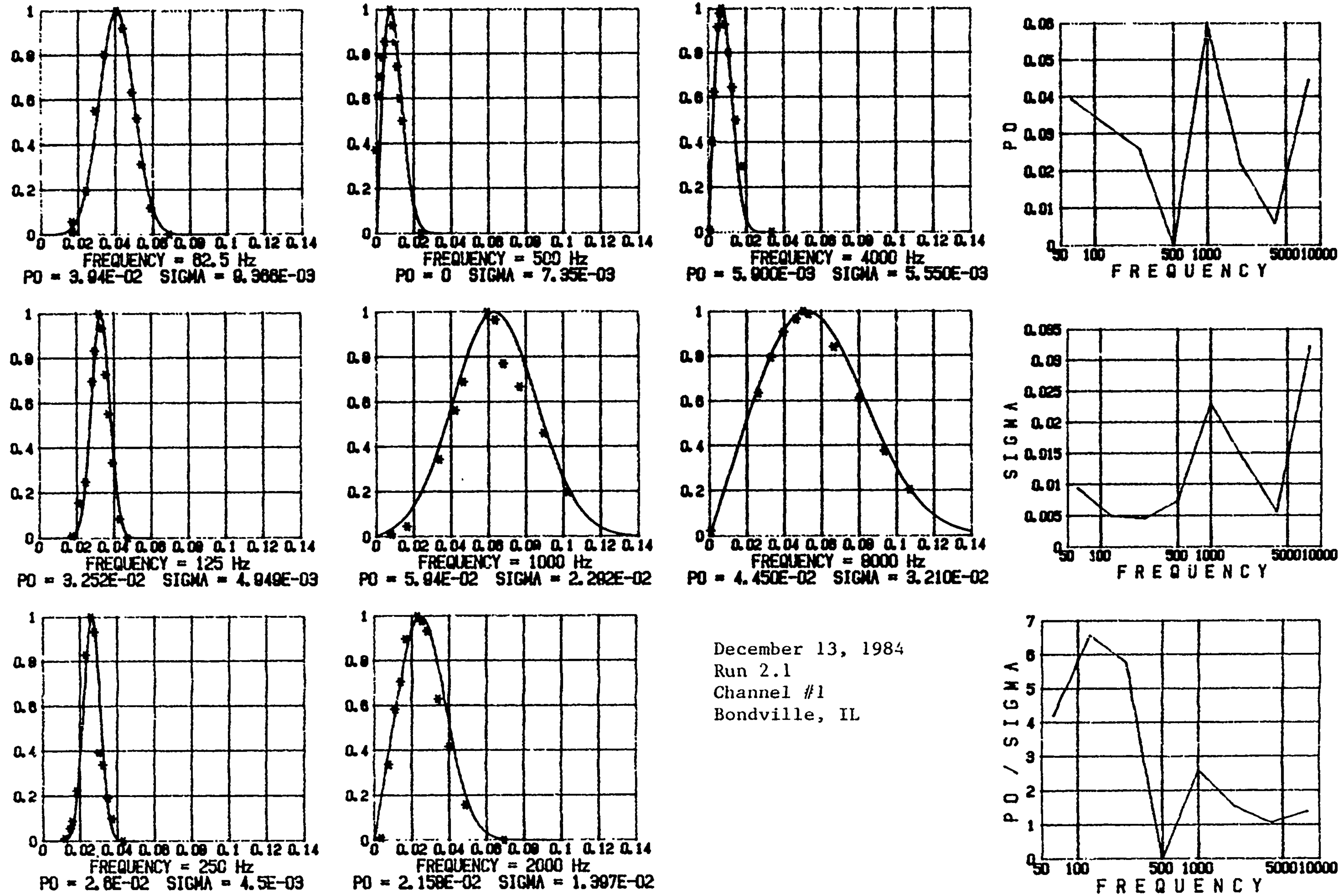
December 13, 1984
Run 1.2
Channel #4
Bondville, IL

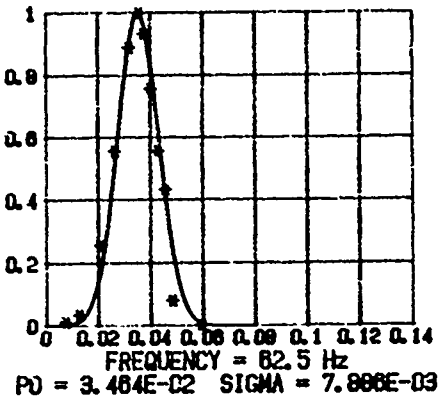


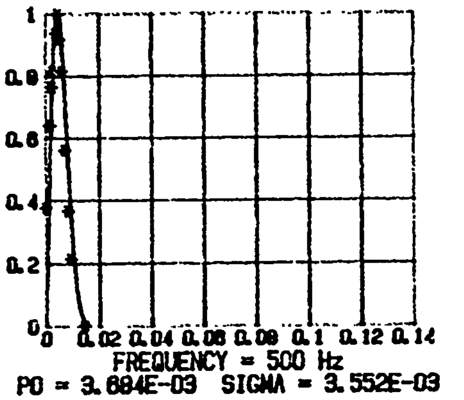


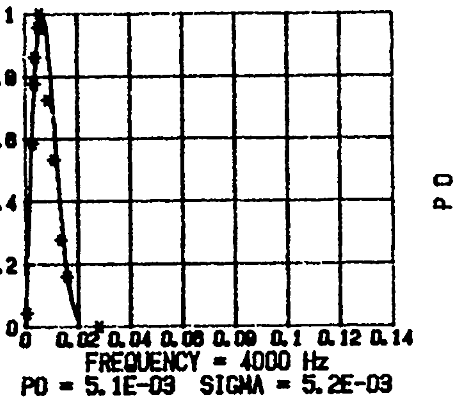
December 13, 1984
Run 1.2
Channel #5
Bondville, IL

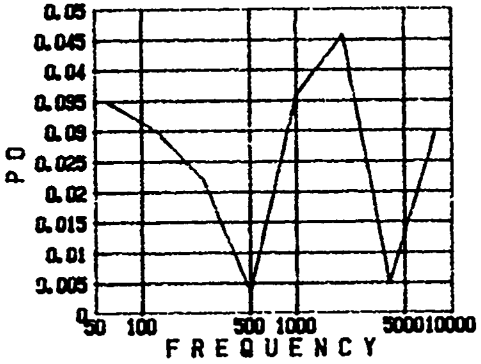


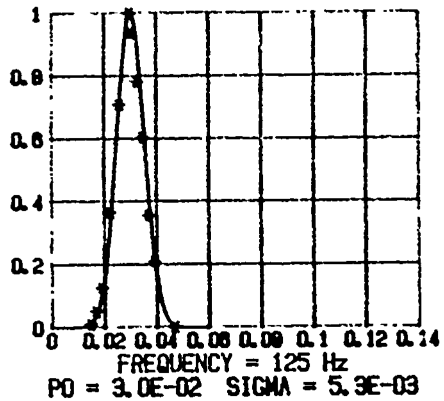

FREQUENCY = 62.5 Hz
PO = 3.94E-02 SIGMA = 9.366E-03
FREQUENCY = 500 Hz
PO = 0 SIGMA = 7.35E-03
FREQUENCY = 4000 Hz
PO = 5.900E-03 SIGMA = 5.550E-03
PO
FREQUENCY
FREQUENCY = 125 Hz
PO = 3.252E-02 SIGMA = 4.949E-03
FREQUENCY = 1000 Hz
PO = 5.94E-02 SIGMA = 2.282E-02
FREQUENCY = 8000 Hz
PO = 4.450E-02 SIGMA = 3.210E-02
SIGMA
FREQUENCY
FREQUENCY = 250 Hz
PO = 2.6E-02 SIGMA = 4.5E-03
FREQUENCY = 2000 Hz
PO = 2.158E-02 SIGMA = 1.397E-02
December 13, 1984
Run 2.1
Channel #1
Bondville, IL
PO / SIGMA
FREQUENCY

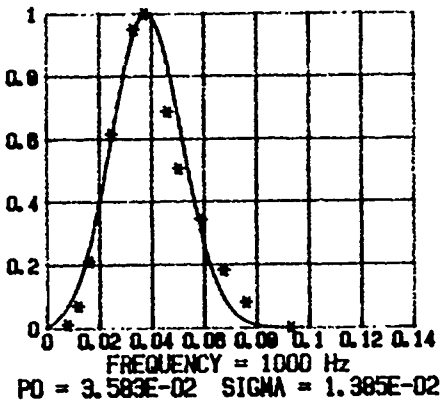


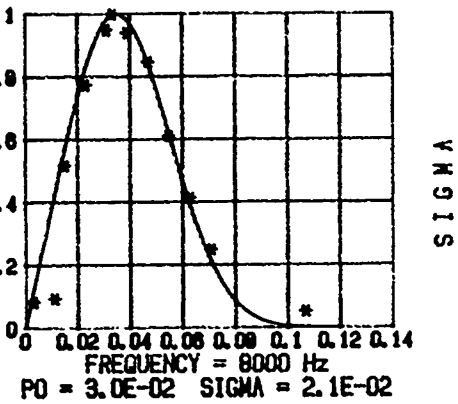


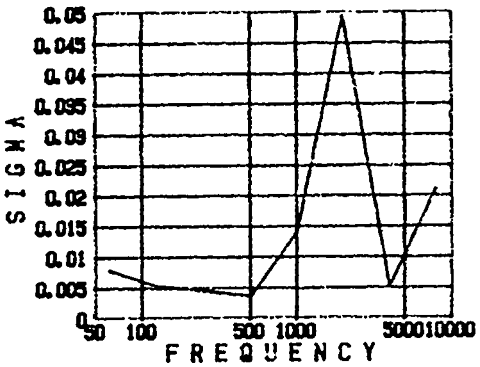


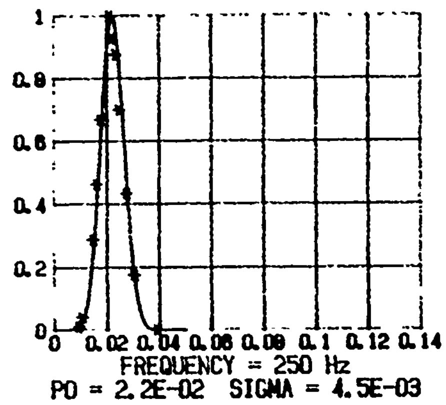


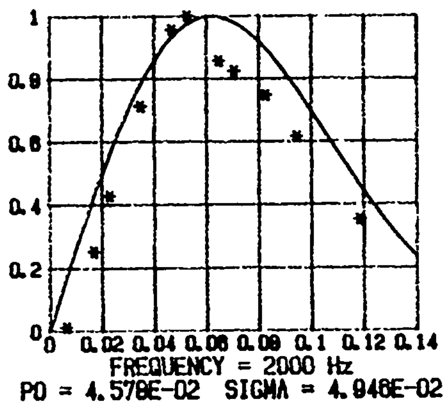




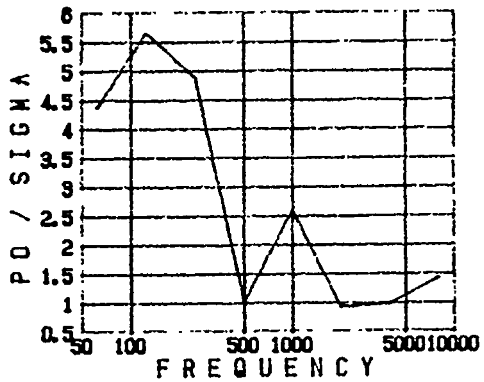


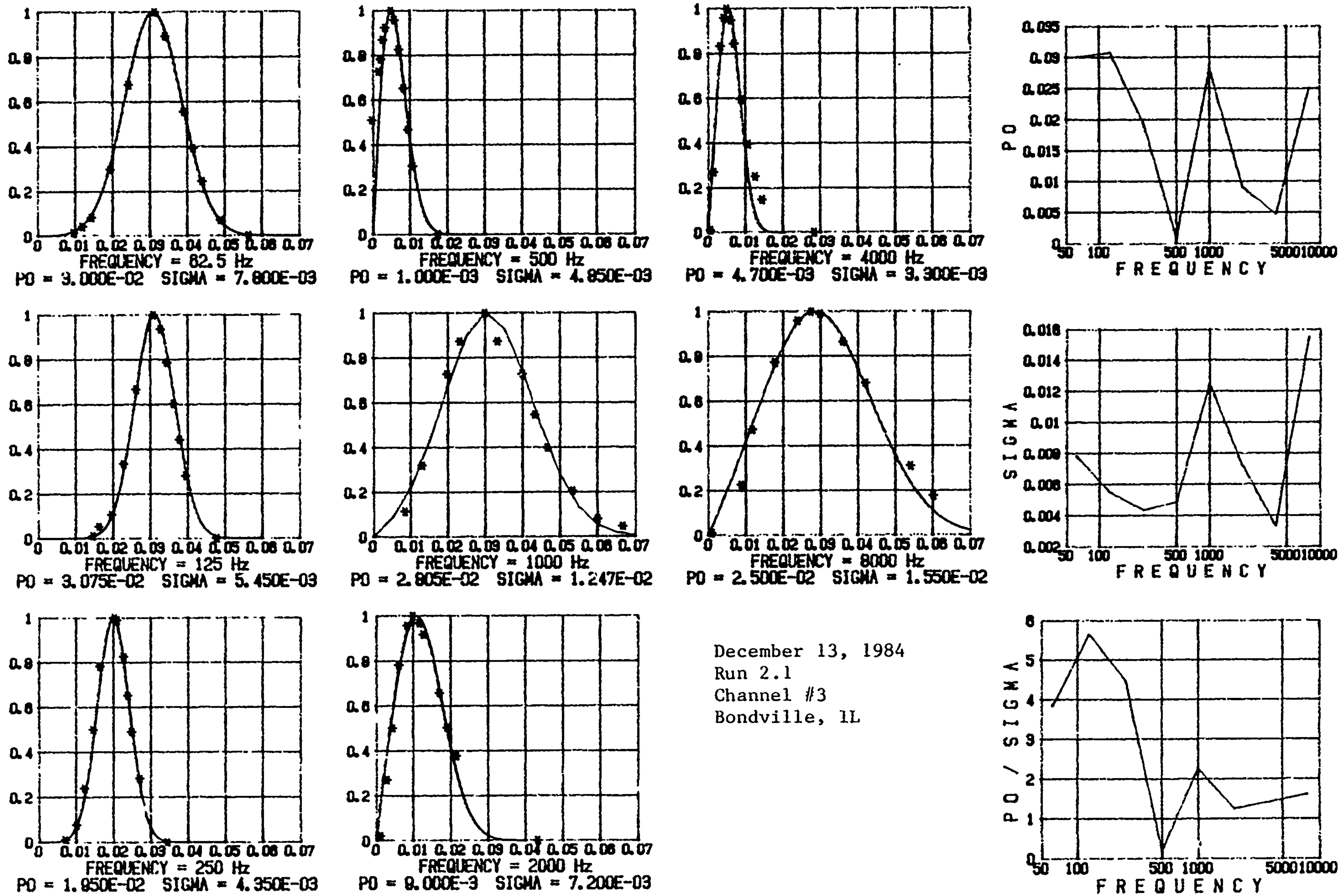




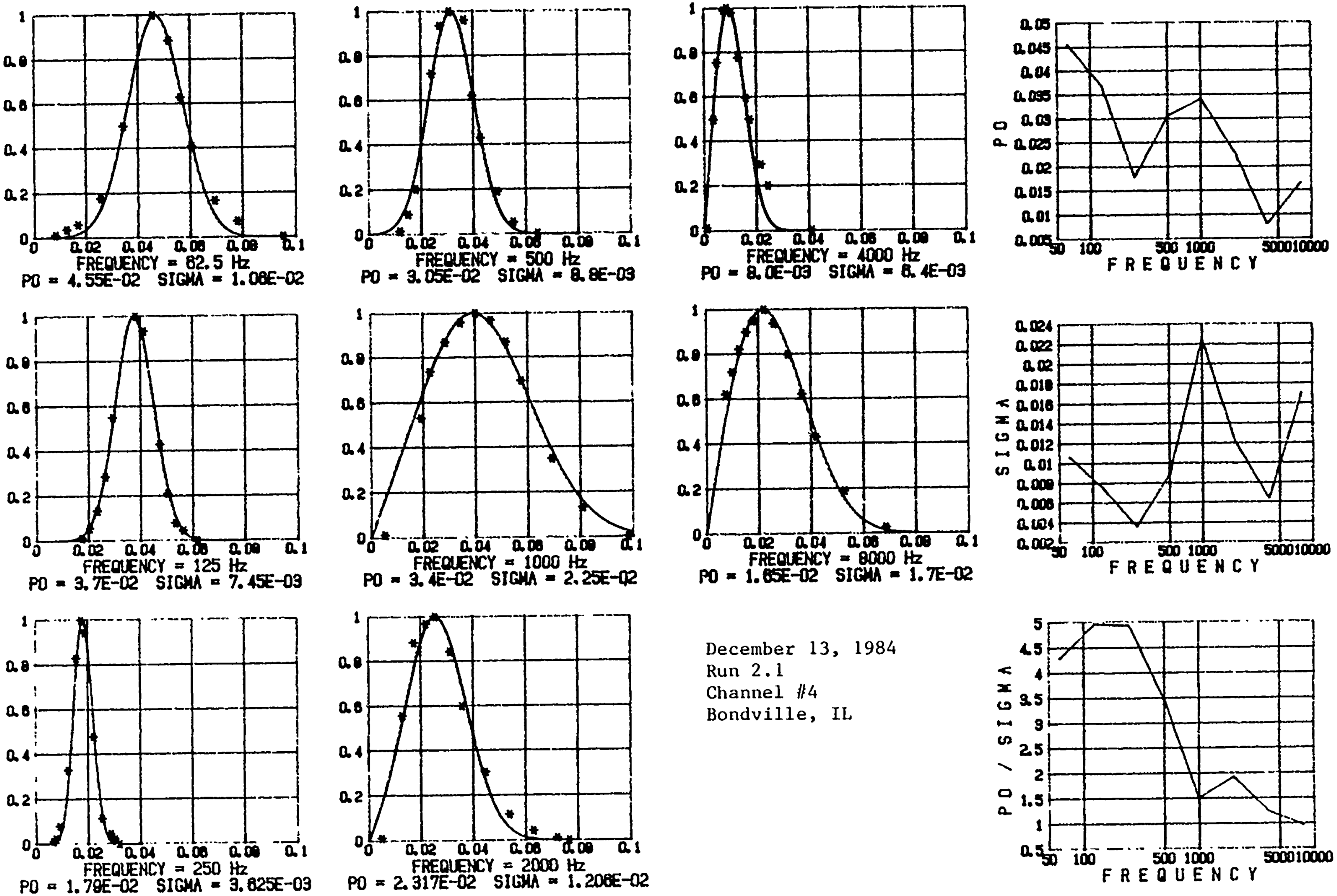


December 13, 1984
Run 2.1
Channel #2
Bondville, IL

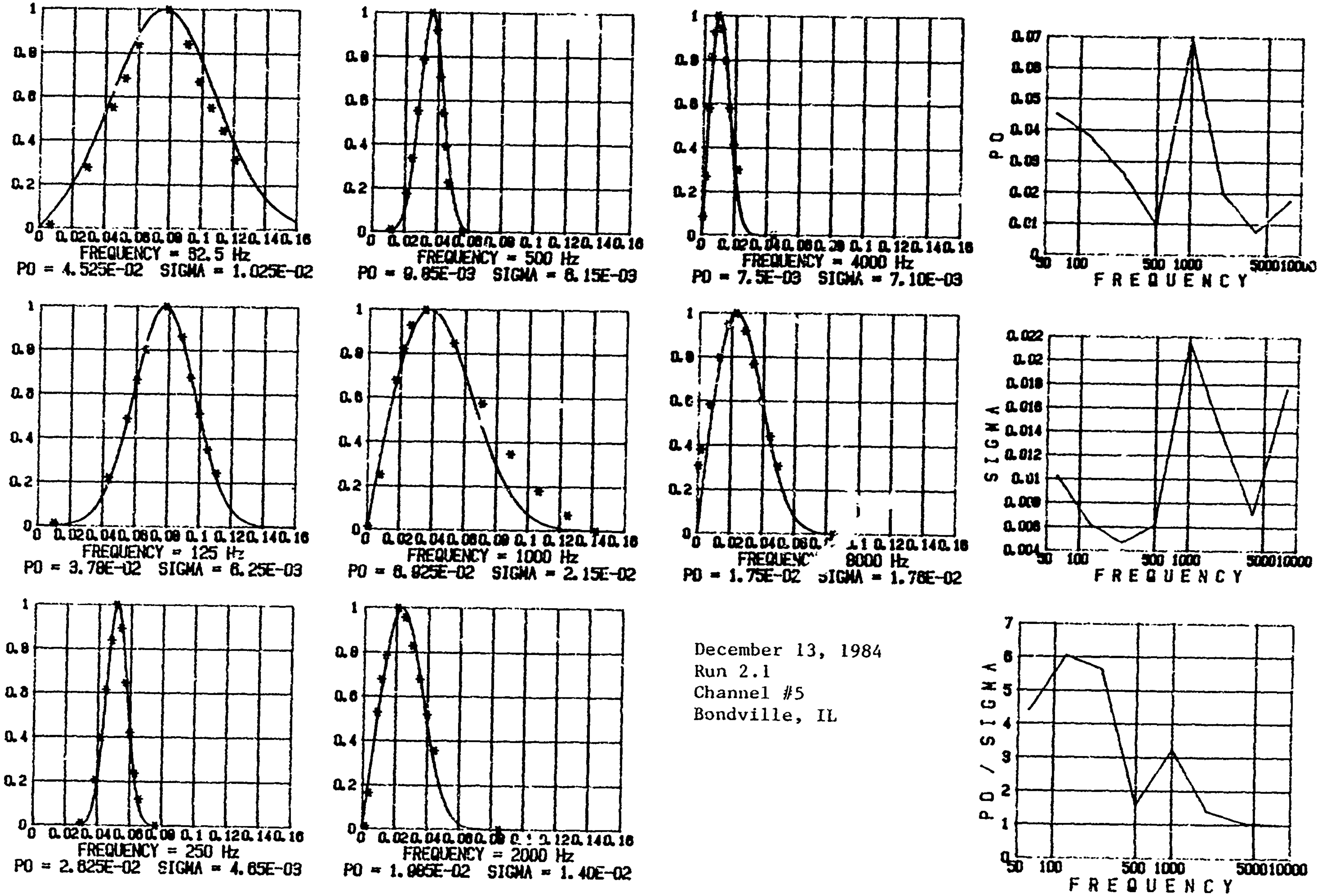




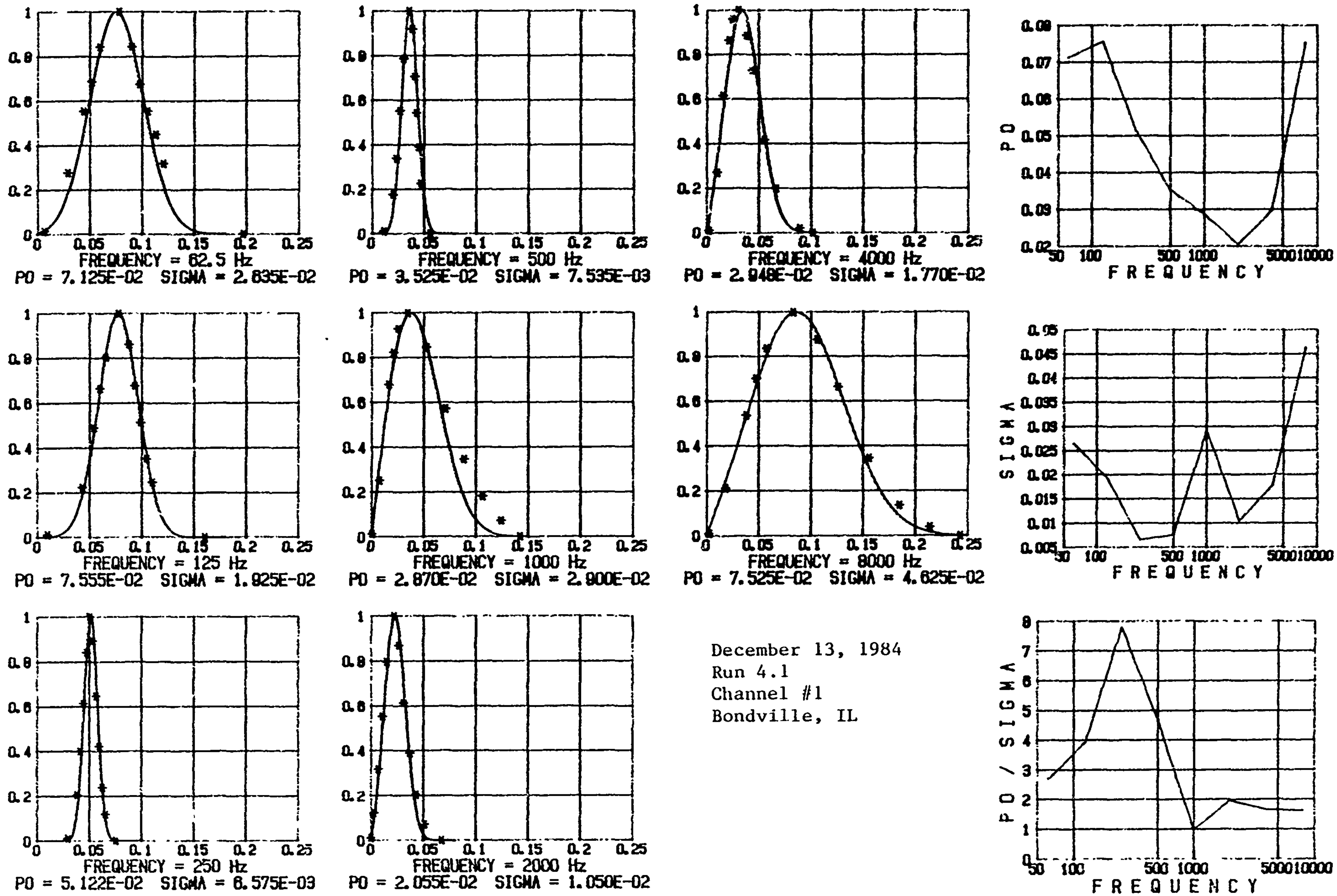
December 13, 1984
Run 2.1
Channel #3
Bondville, IL

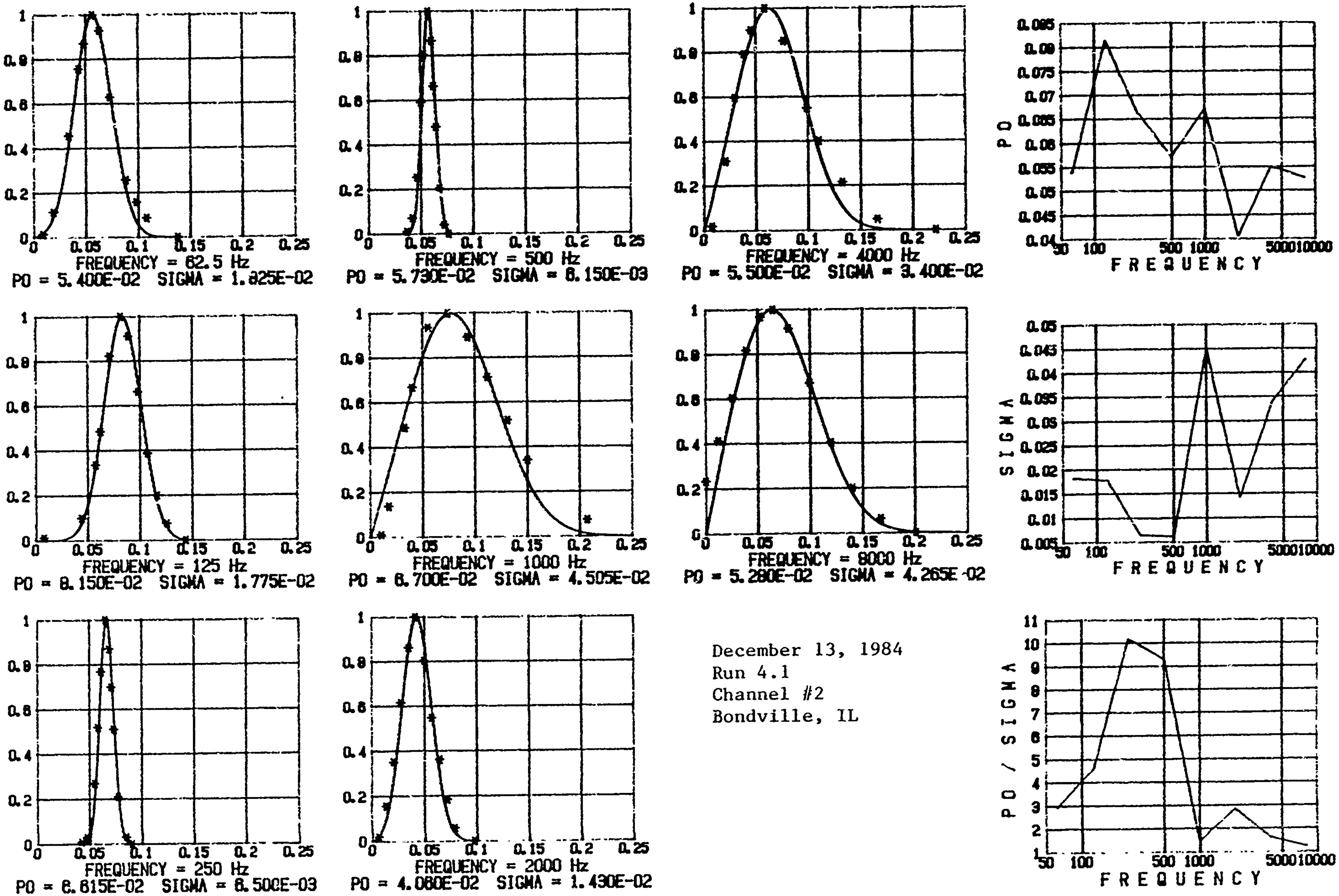


December 13, 1984
Run 2.1
Channel #4
Bondville, IL

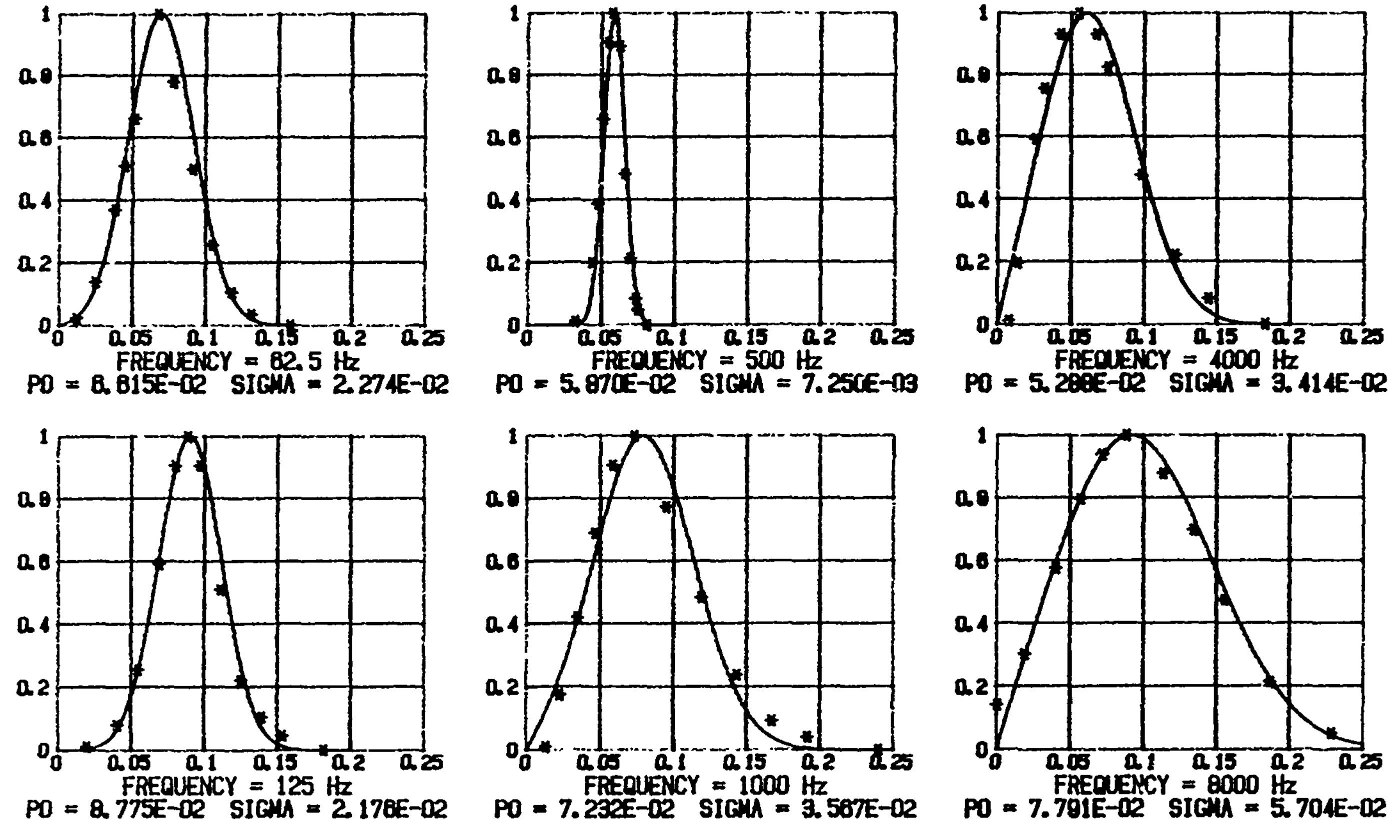


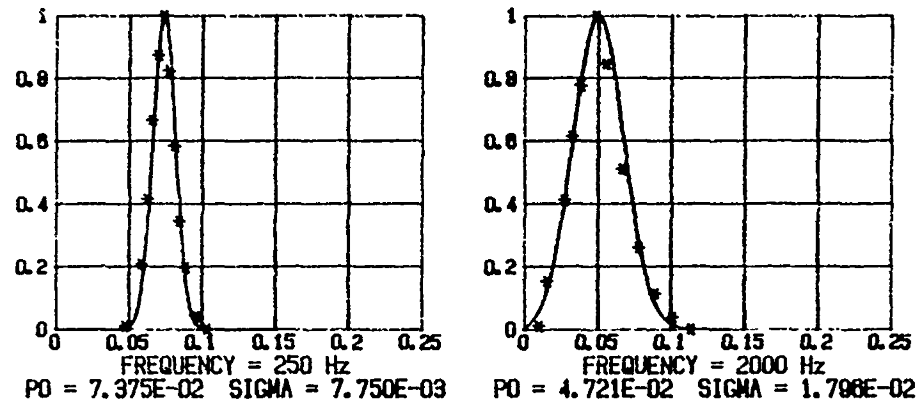
December 13, 1984
Run 2.1
Channel #5
Bondville, IL


FREQUENCY = 62.5 Hz
PO = 7.125E-02 SIGMA = 2.835E-02
FREQUENCY = 500 Hz
PO = 3.525E-02 SIGMA = 7.535E-03
FREQUENCY = 4000 Hz
PO = 2.948E-02 SIGMA = 1.770E-02
PO
FREQUENCY
FREQUENCY = 125 Hz
PO = 7.555E-02 SIGMA = 1.825E-02
FREQUENCY = 1000 Hz
PO = 2.870E-02 SIGMA = 2.900E-02
FREQUENCY = 8000 Hz
PO = 7.525E-02 SIGMA = 4.625E-02
SIGMA
FREQUENCY
FREQUENCY = 250 Hz
PO = 5.122E-02 SIGMA = 6.575E-03
FREQUENCY = 2000 Hz
PO = 2.055E-02 SIGMA = 1.050E-02
December 13, 1984
Run 4.1
Channel #1
Bondville, IL
PO / SIGMA
FREQUENCY

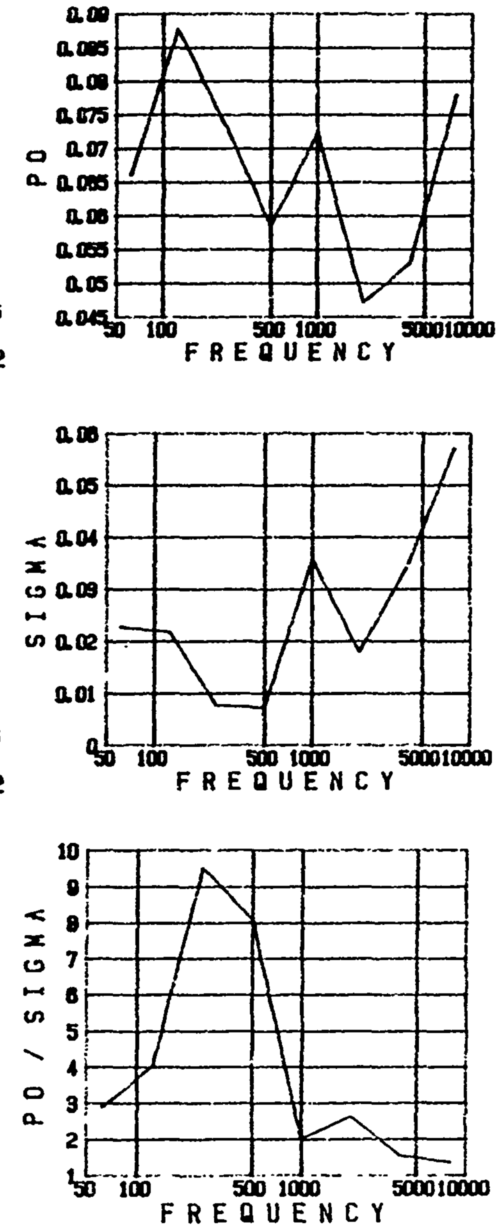


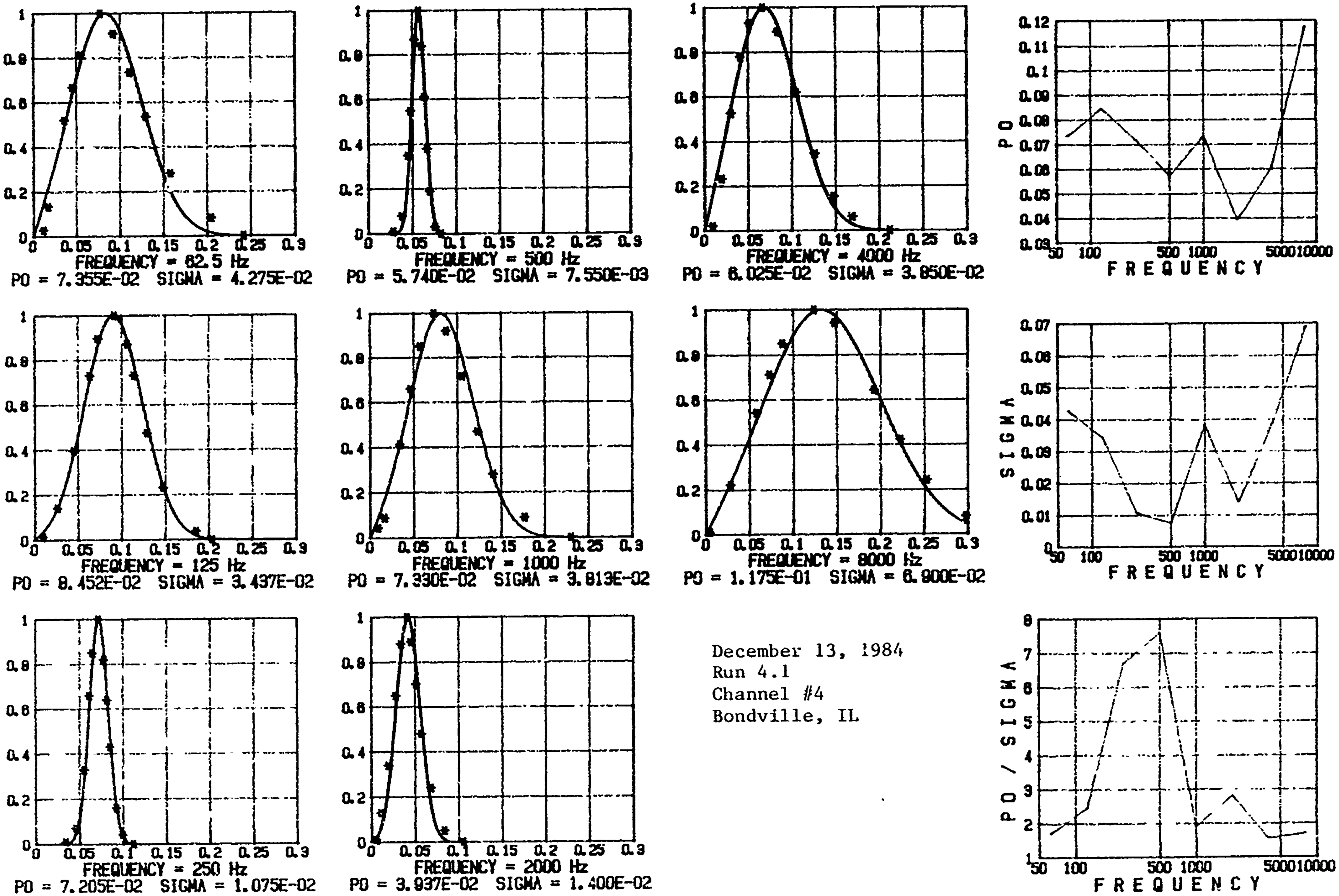
December 13, 1984
Run 4.1
Channel #2
Bondville, IL



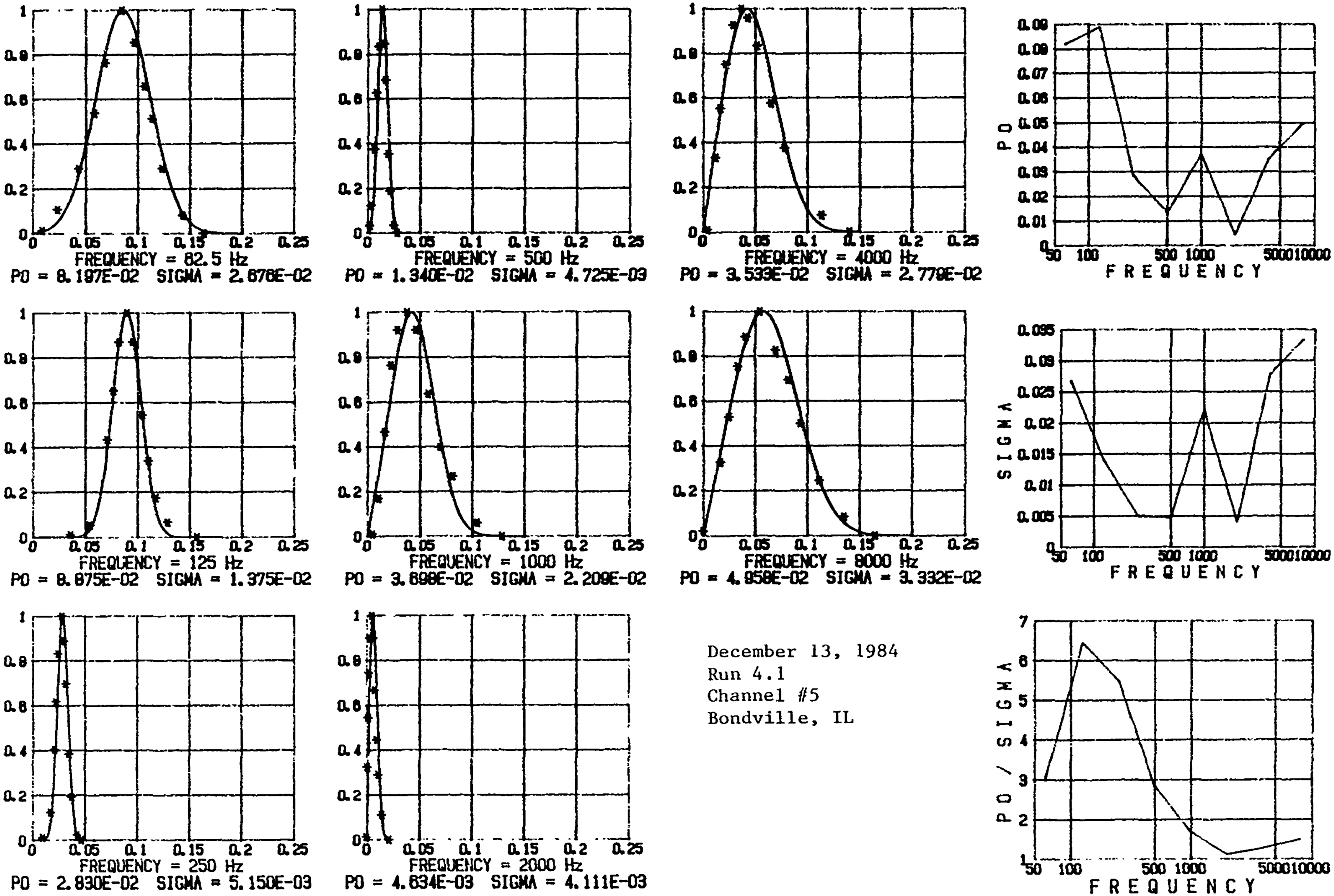


December 13, 1984
Run 4.1
Channel #3
Bondville, IL

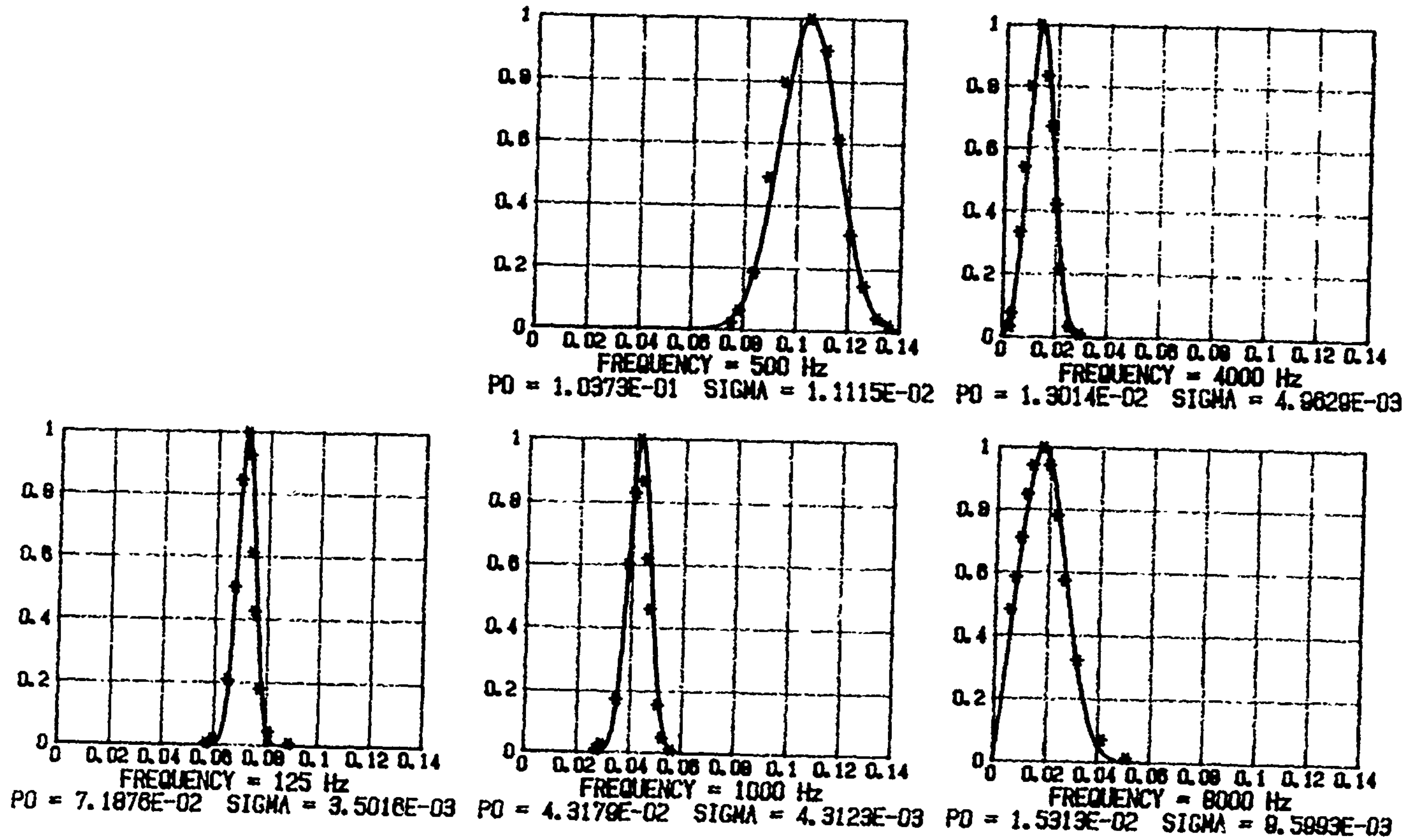


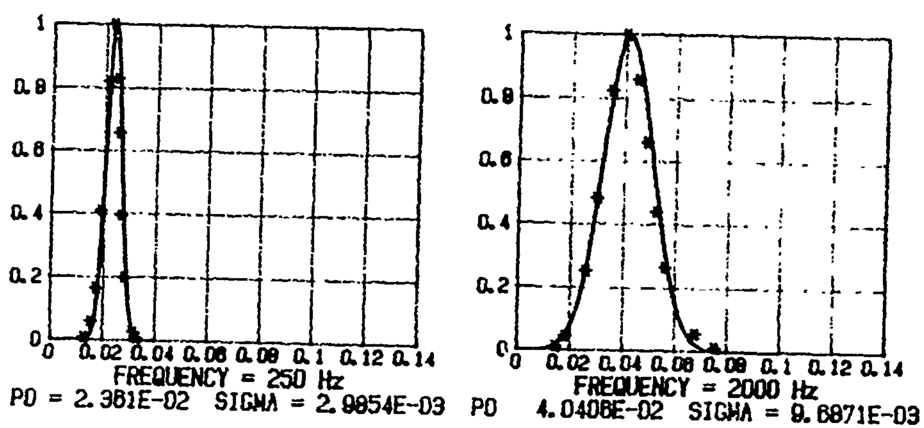


December 13, 1984
Run 4.1
Channel #4
Bondville, IL

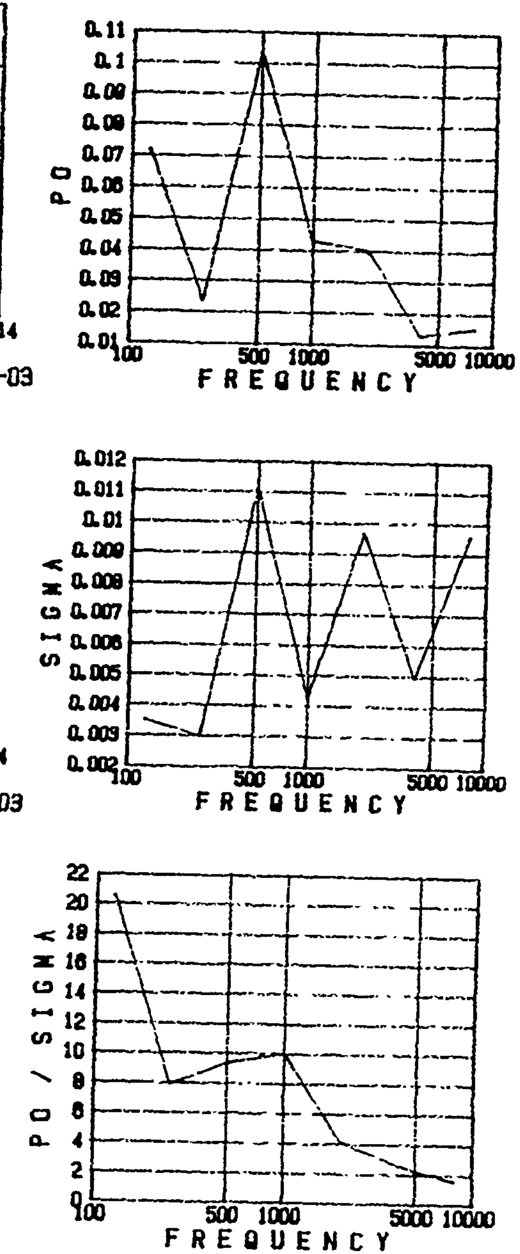


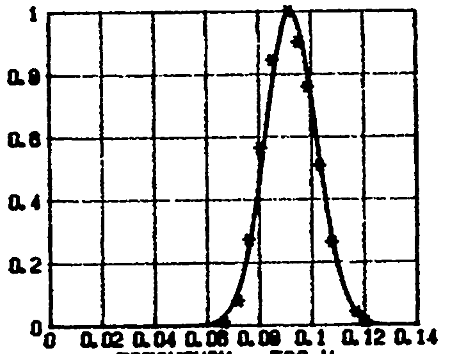
December 13, 1984
Run 4.1
Channel #5
Bondville, IL

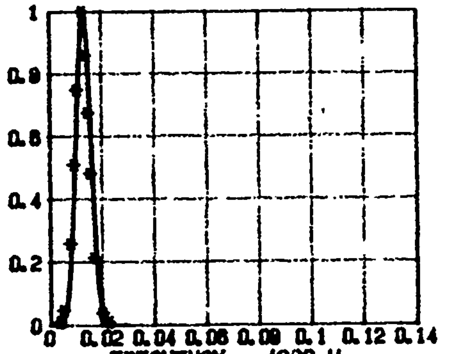


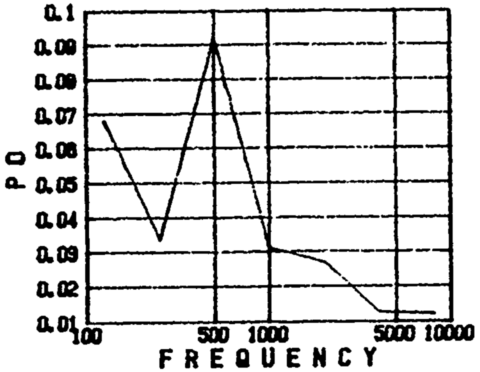


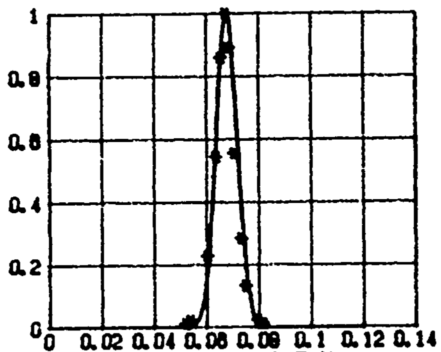
January 11, 1985
Run 2.1
Channel #1
Bondville, IL

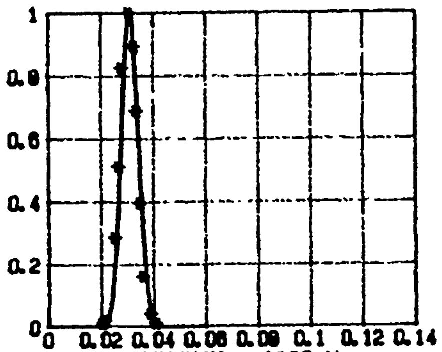


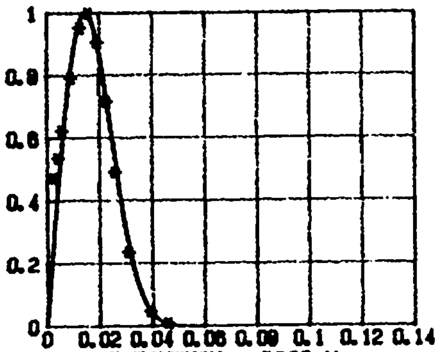

FREQUENCY = 500 Hz
PO = 9.1712E-02 SIGMA = 9.7454E-03

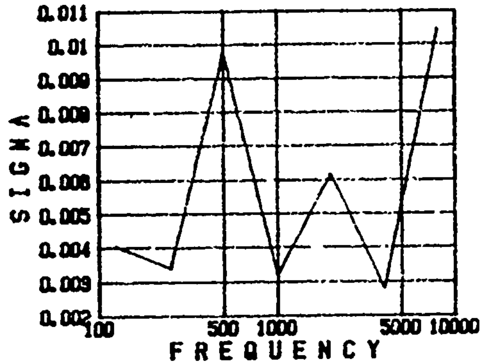

FREQUENCY = 4000 Hz
PO = 1.2889E-02 SIGMA = 2.8125E-03

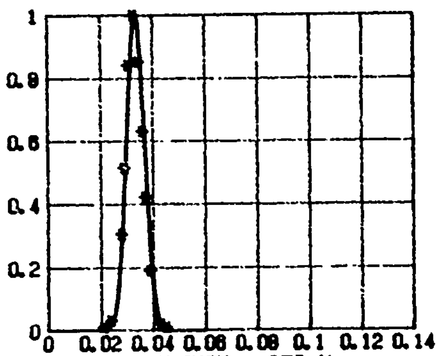


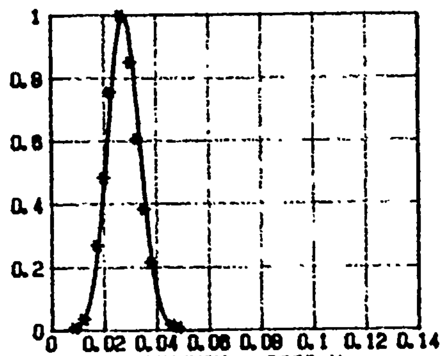

FREQUENCY = 125 Hz
PO = 6.7992E-02 SIGMA = 4.0344E-03


FREQUENCY = 1000 Hz
PO = 3.1153E-02 SIGMA = 3.2014E-03

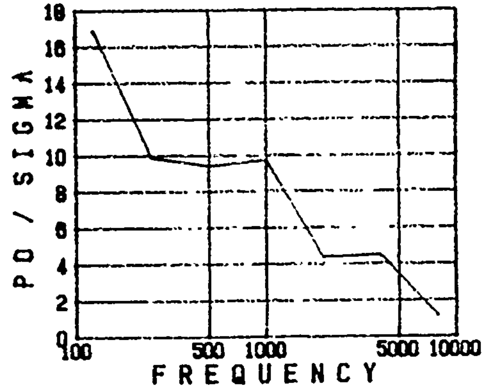

FREQUENCY = 8000 Hz
PO = 1.216E-02 SIGMA = 1.0429E-02

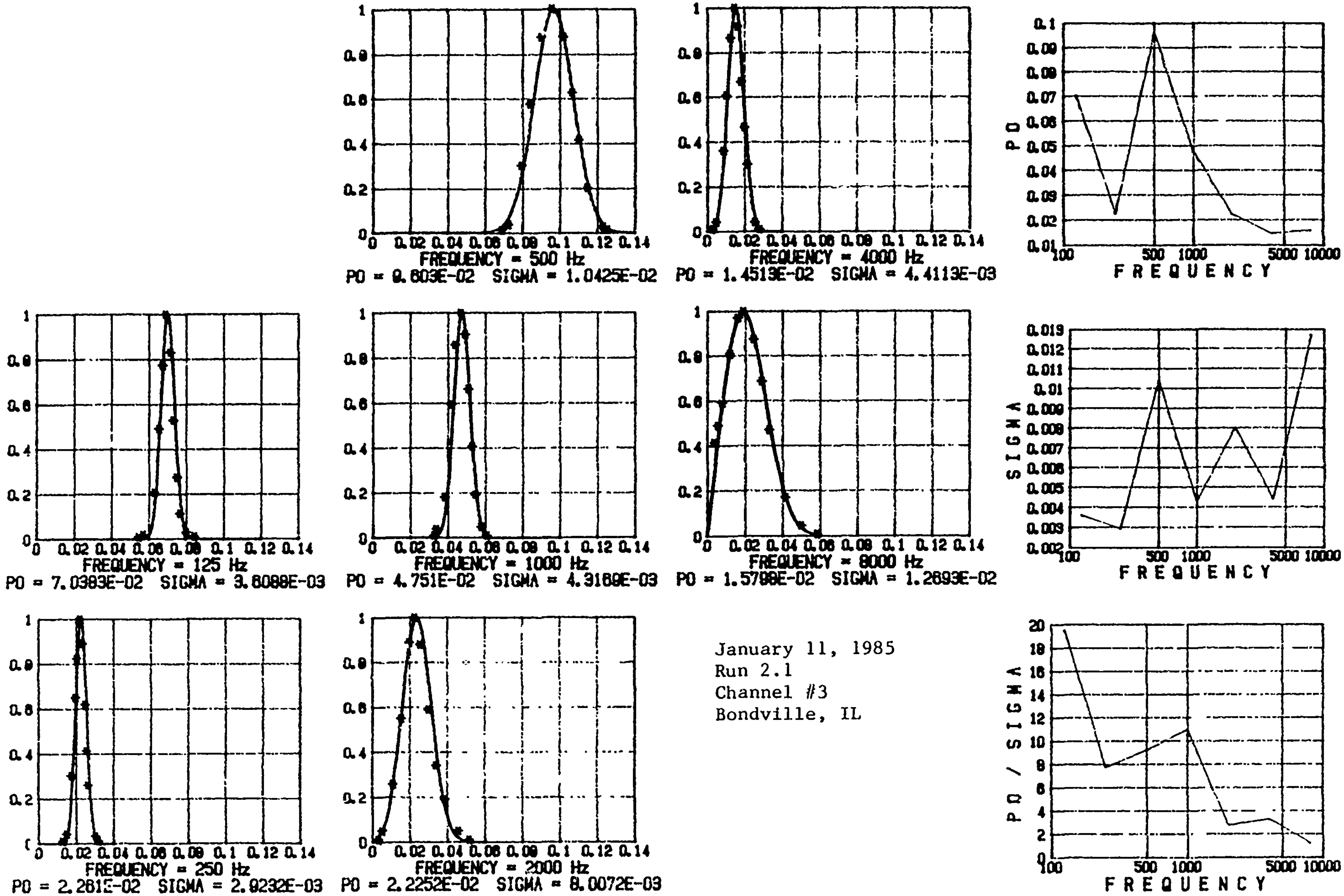



FREQUENCY = 250 Hz
PO = 3.3686E-02 SIGMA = 3.4015E-03

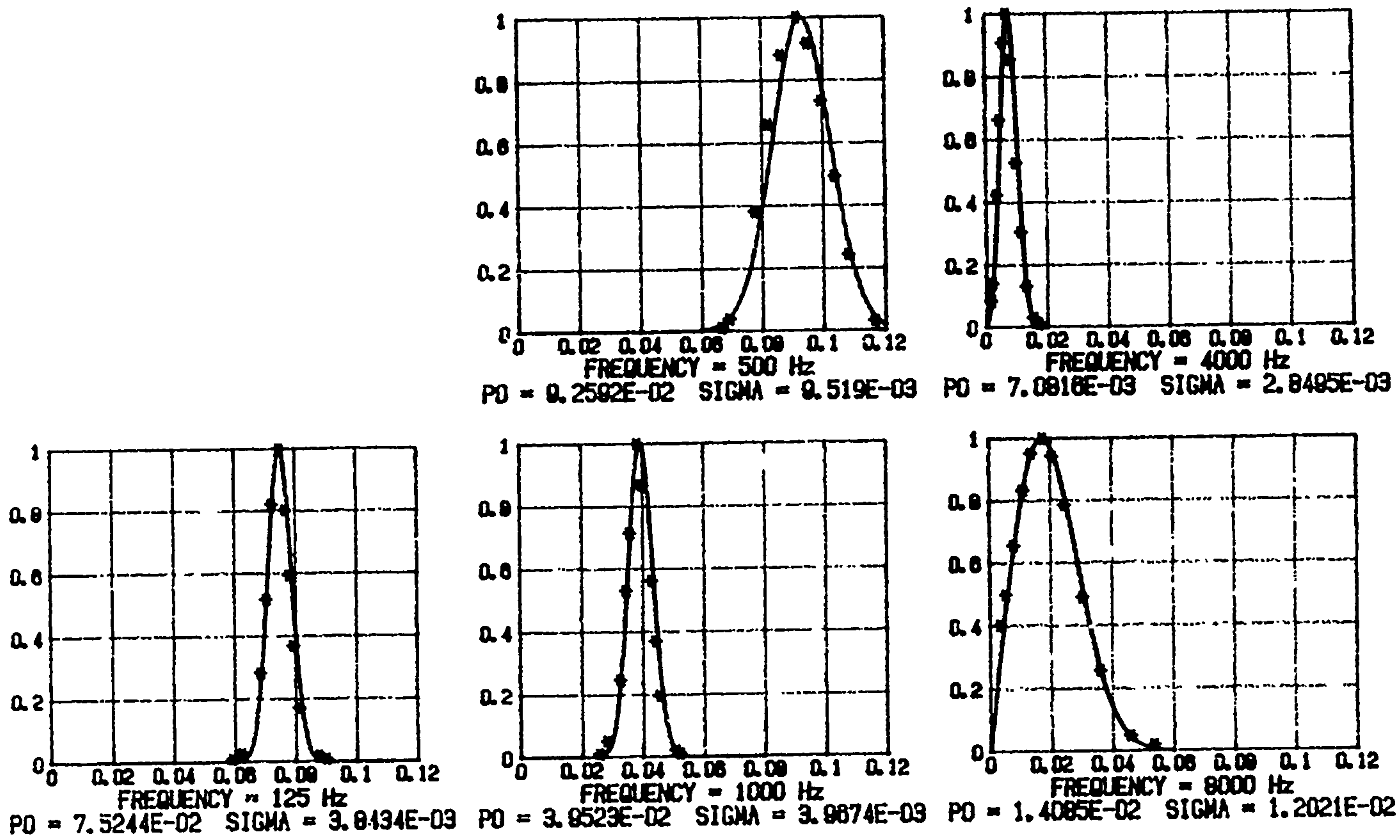

FREQUENCY = 2000 Hz
PO = 2.6918E-02 SIGMA = 6.1715E-03

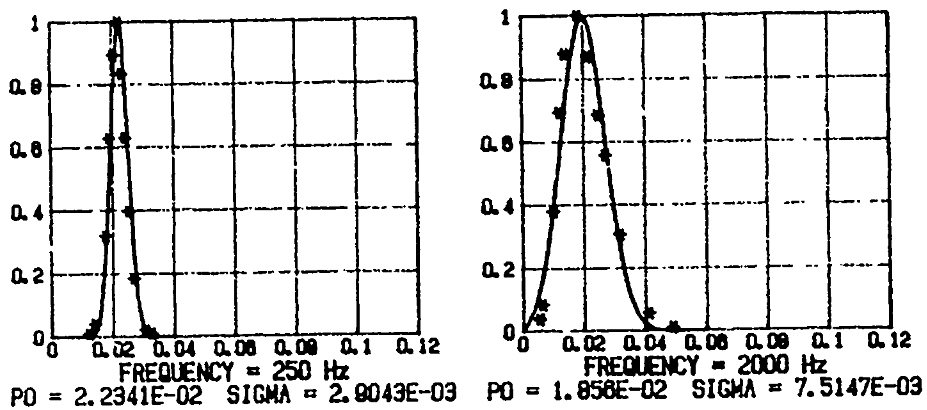
January 11, 1985
Run 2.1
Channel #2
Bondville, IL



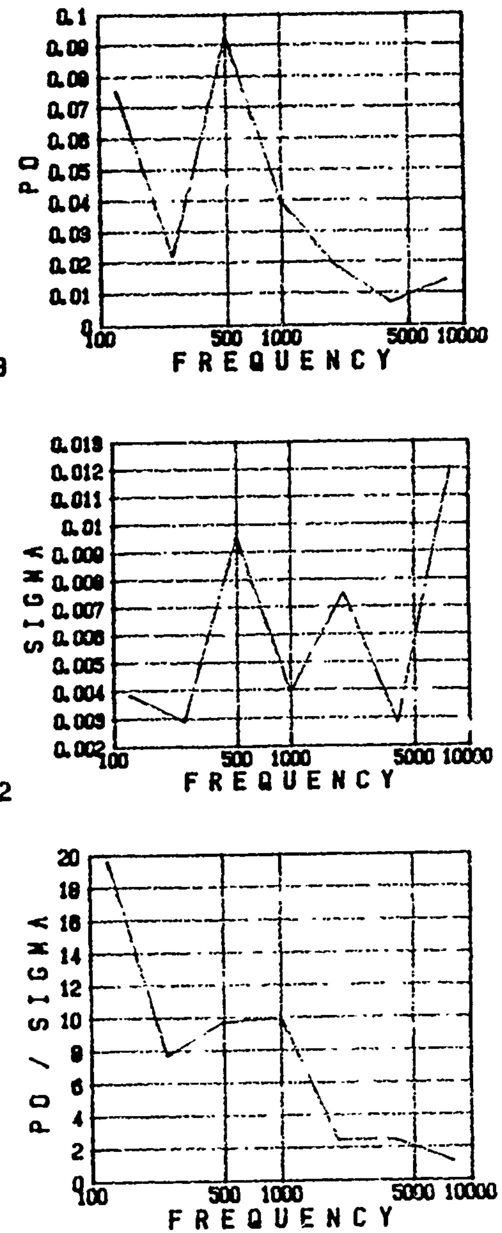


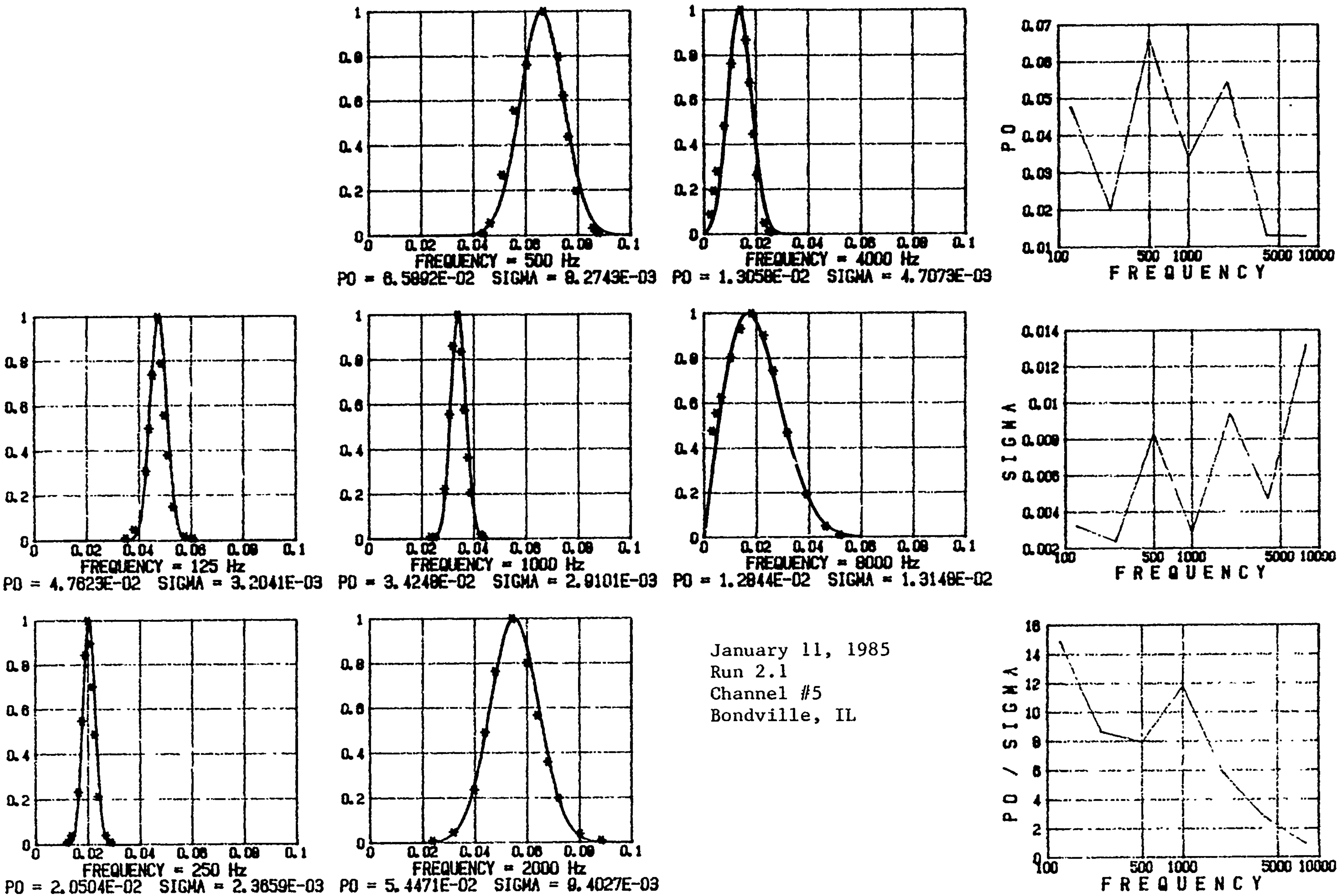
January 11, 1985
Run 2.1
Channel #3
Bondville, IL



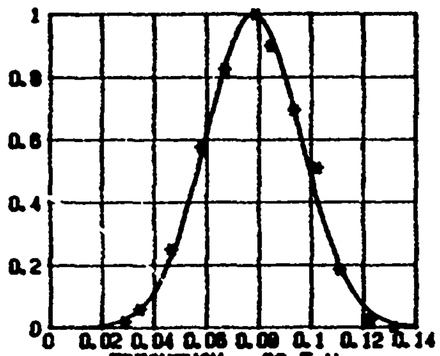


January 11, 1985
Run 2.1
Channel #4
Bondville, IL

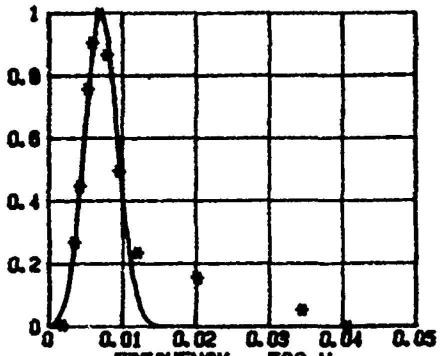




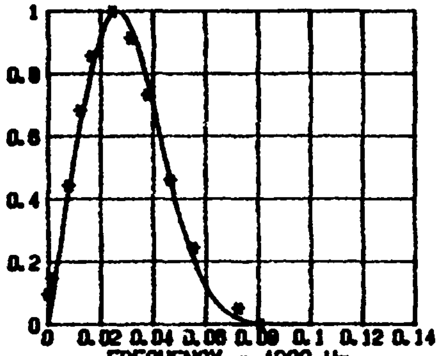
January 11, 1985
Run 2.1
Channel #5
Bondville, IL



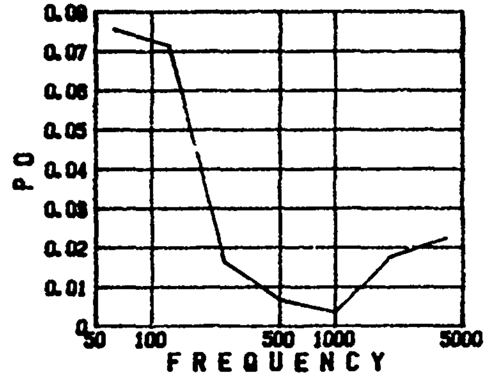
FREQUENCY = 62.5 Hz
PO = 7.570E-02 SIGMA = 1.855E-02

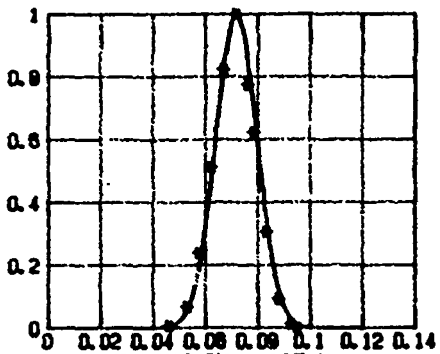


FREQUENCY = 500 Hz
PO = 6.750E-03 SIGMA = 2.300E-03

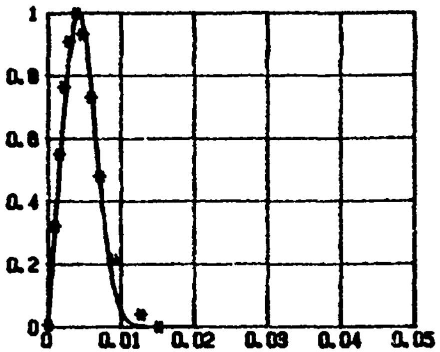


FREQUENCY = 4000 Hz
PO = 2.240E-02 SIGMA = 1.690E-02

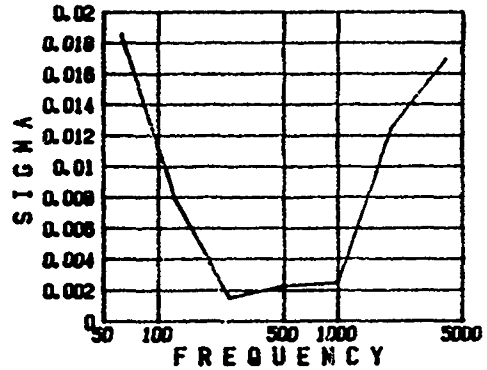


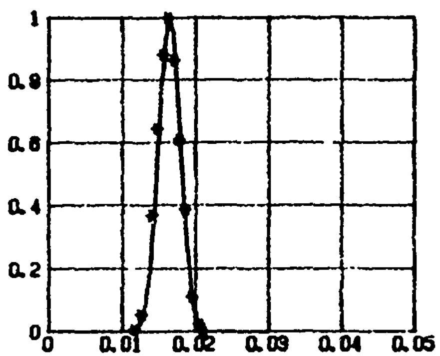


FREQUENCY = 125 Hz
PO = 7.155E-02 SIGMA = 7.887E-03

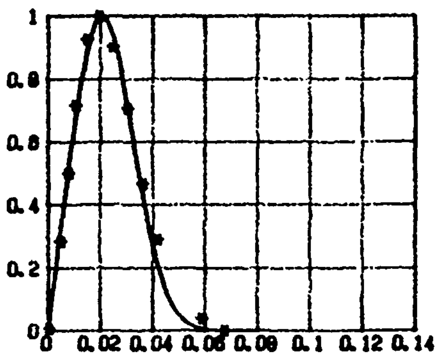


FREQUENCY = 1000 Hz
PO = 3.600E-03 SIGMA = 2.450E-03



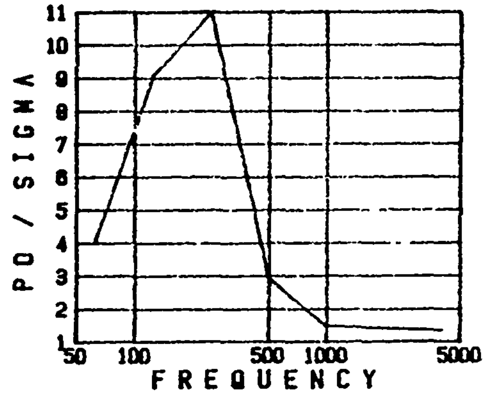


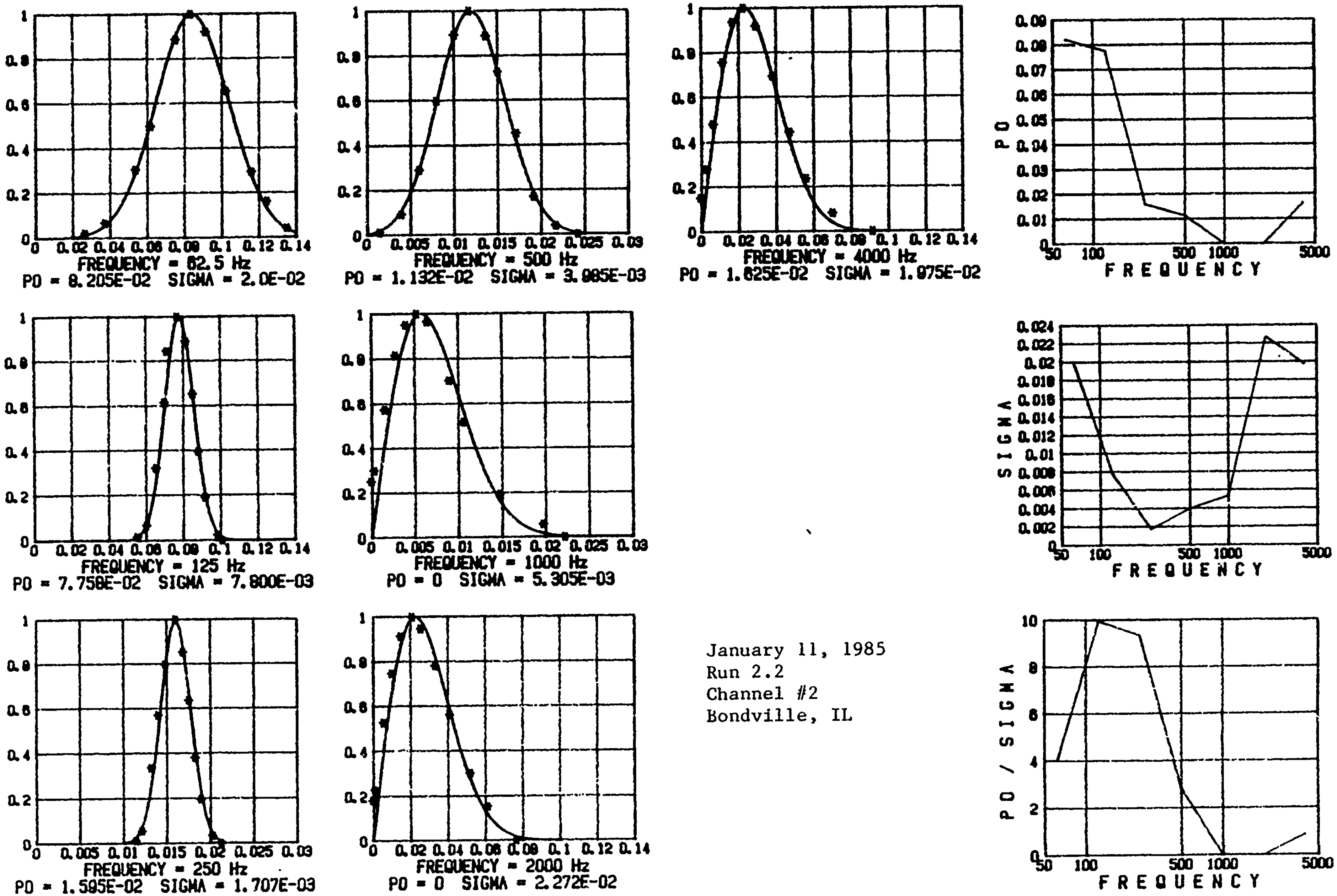
FREQUENCY = 250 Hz
PO = 1.640E-02 SIGMA = 1.500E-03



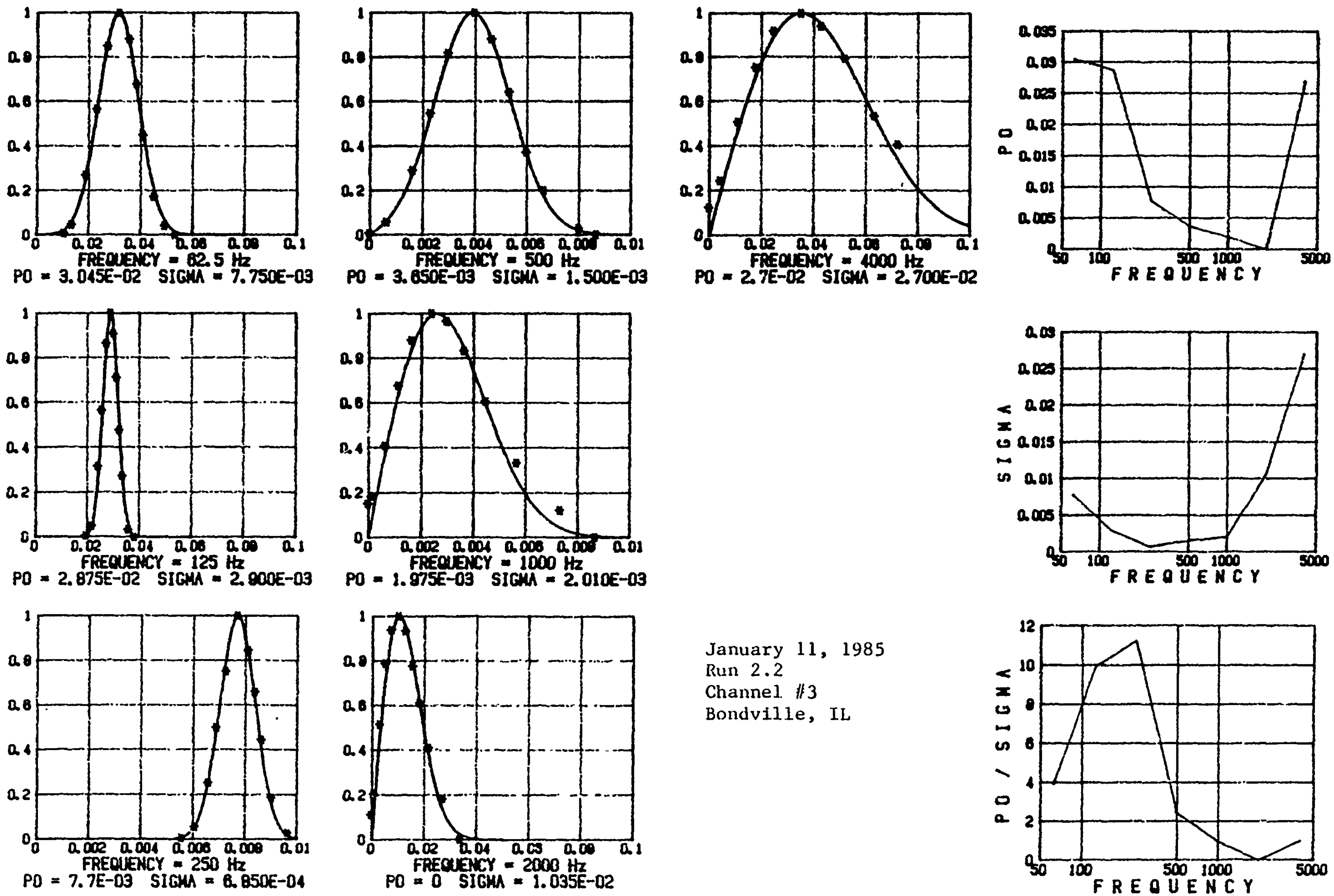
FREQUENCY = 2000 Hz
PO = 1.750E-02 SIGMA = 1.250E-02

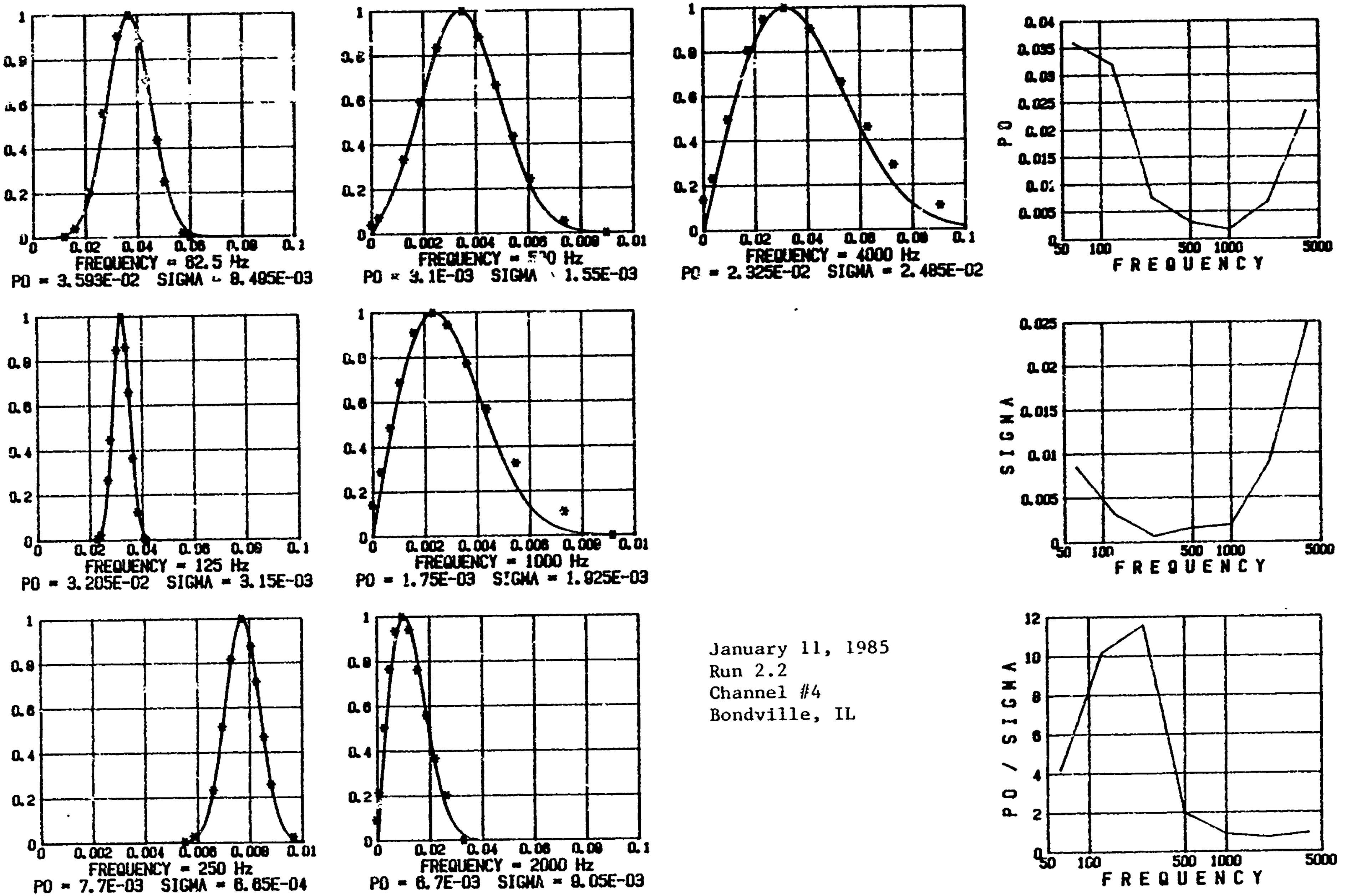
January 11, 1985
Run 2.2
Channel #1
Bondville, IL



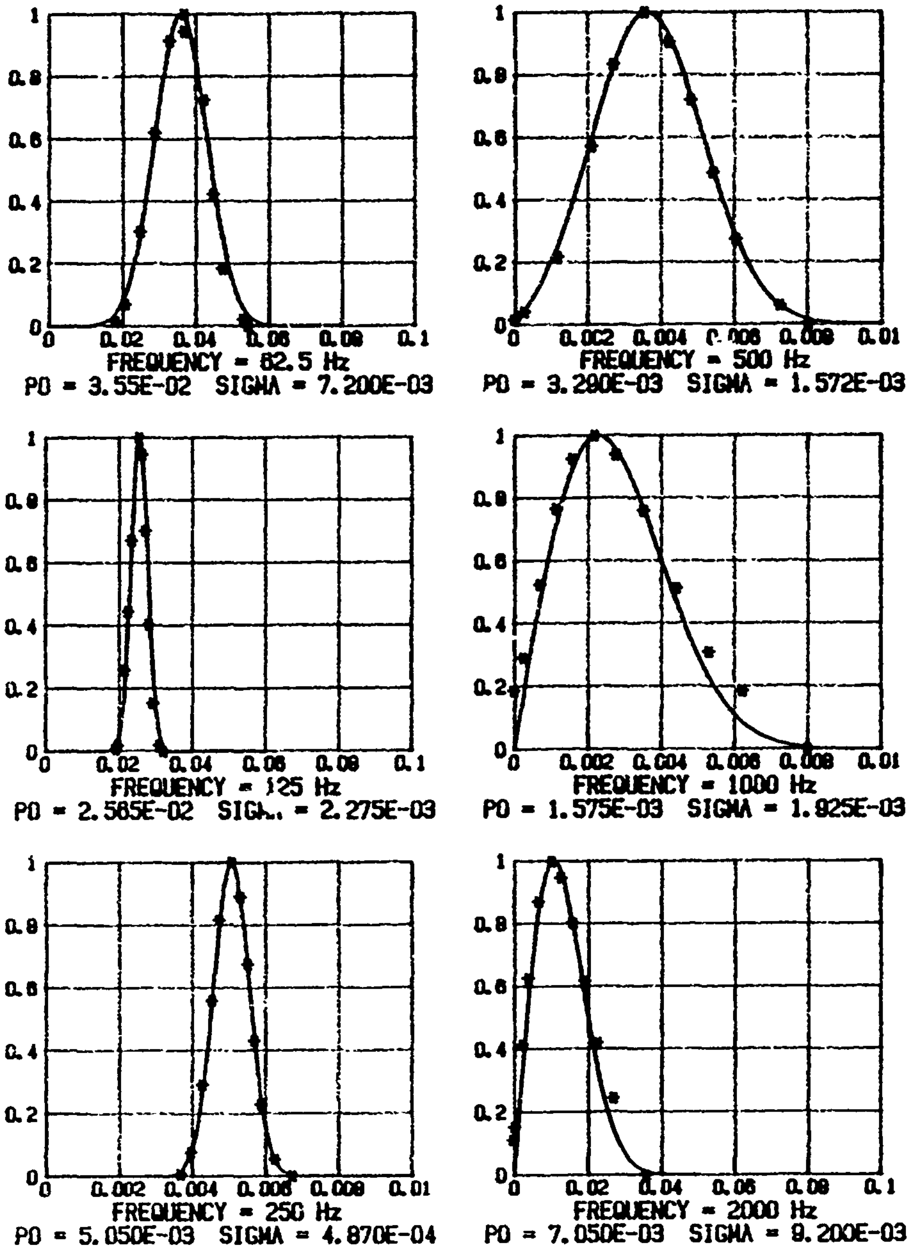


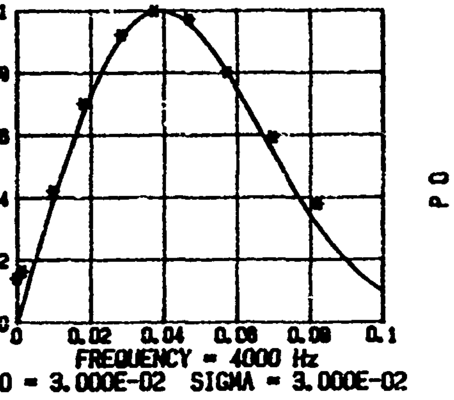
January 11, 1985
Run 2.2
Channel #2
Bondville, IL

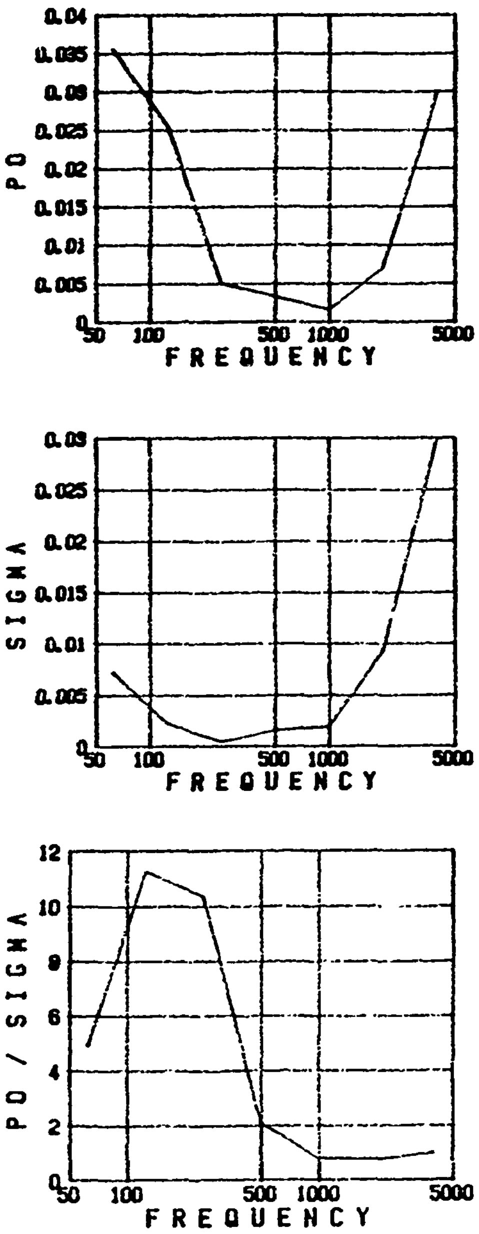

FREQUENCY = 62.5 Hz
PO = 3.045E-02 SIGMA = 7.750E-03
FREQUENCY = 500 Hz
PO = 3.650E-03 SIGMA = 1.500E-03
FREQUENCY = 4000 Hz
PO = 2.7E-02 SIGMA = 2.700E-02
PO
FREQUENCY
FREQUENCY = 125 Hz
PO = 2.875E-02 SIGMA = 2.900E-03
FREQUENCY = 1000 Hz
PO = 1.975E-03 SIGMA = 2.010E-03
SIGMA
FREQUENCY
FREQUENCY = 250 Hz
PO = 7.7E-03 SIGMA = 6.850E-04
FREQUENCY = 2000 Hz
PO = 0 SIGMA = 1.035E-02
January 11, 1985
Run 2.2
Channel #3
Bondville, IL
PO / SIGMA
FREQUENCY



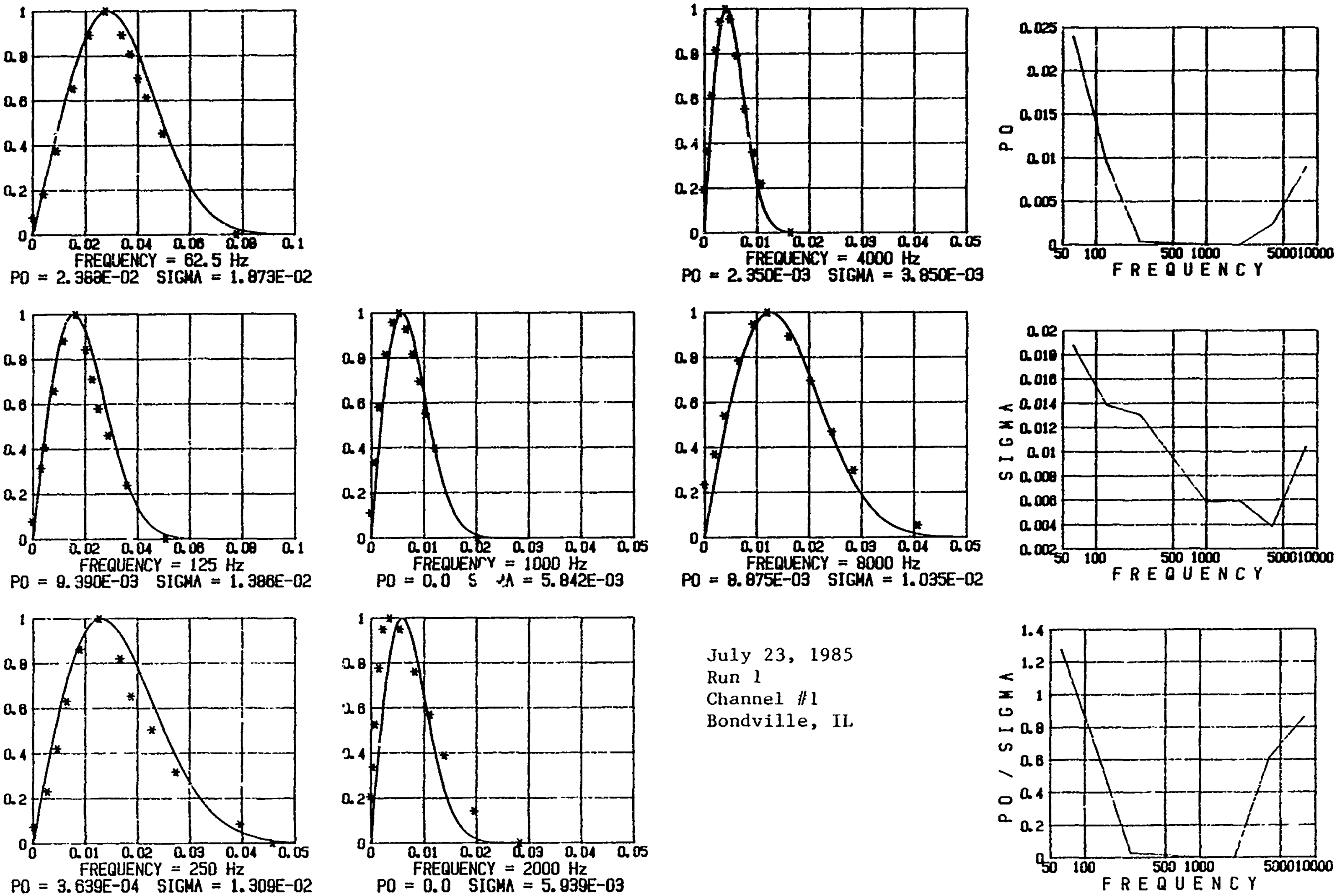
January 11, 1985
Run 2.2
Channel #4
Bondville, IL



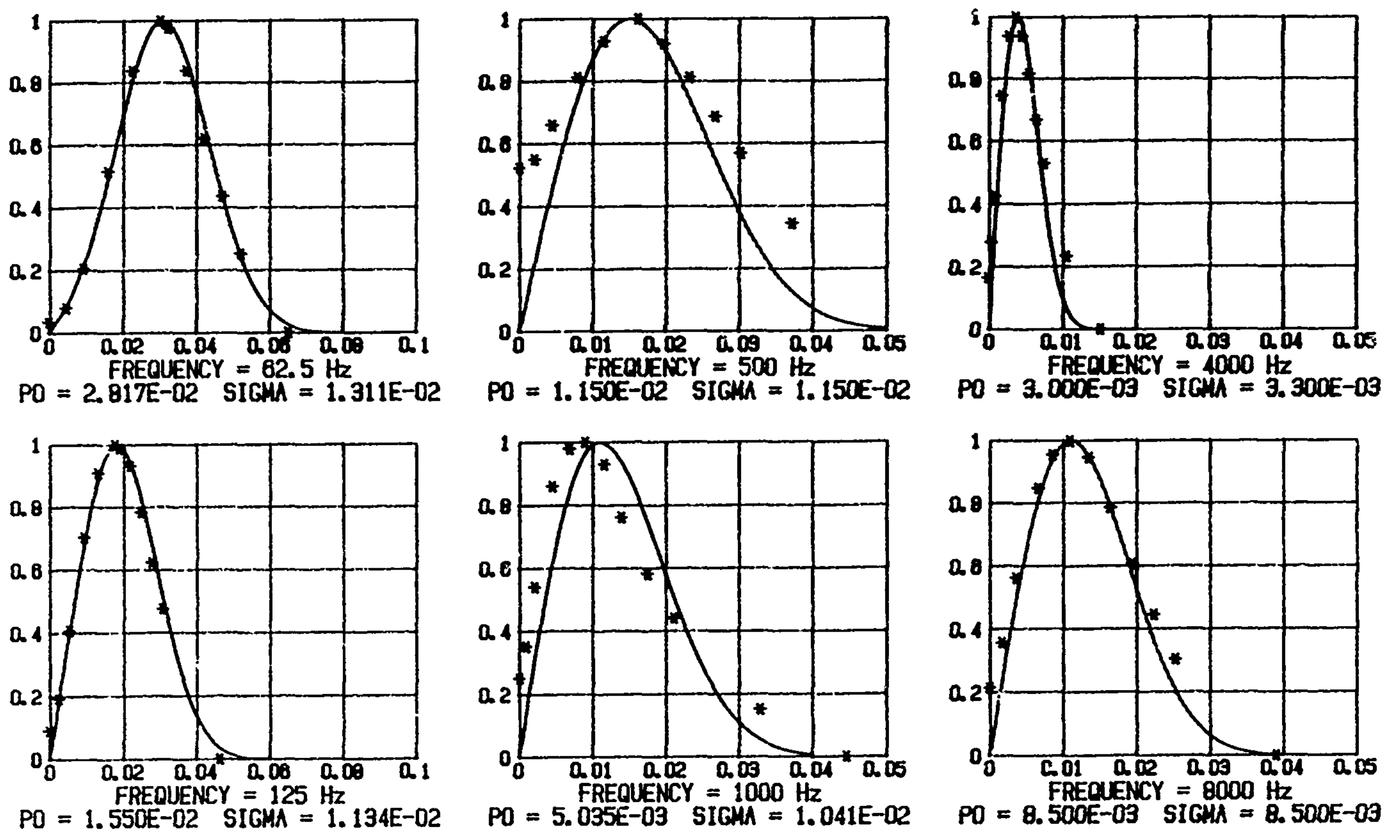


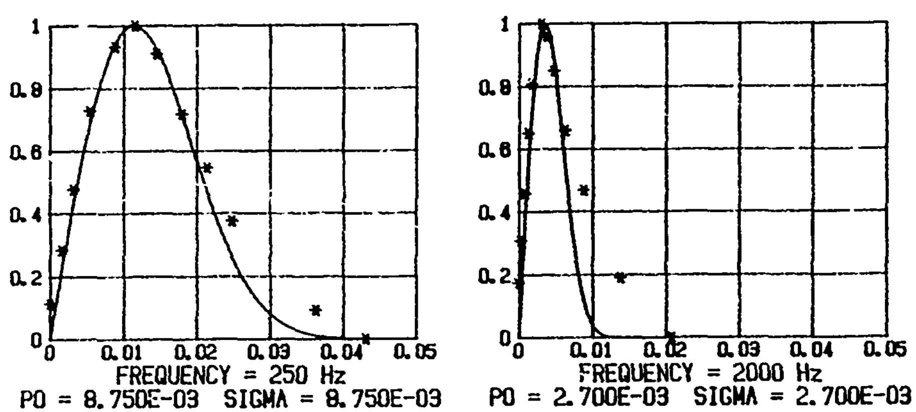


January 11, 1985
Run 2.2
Channel #5
Bondville, IL

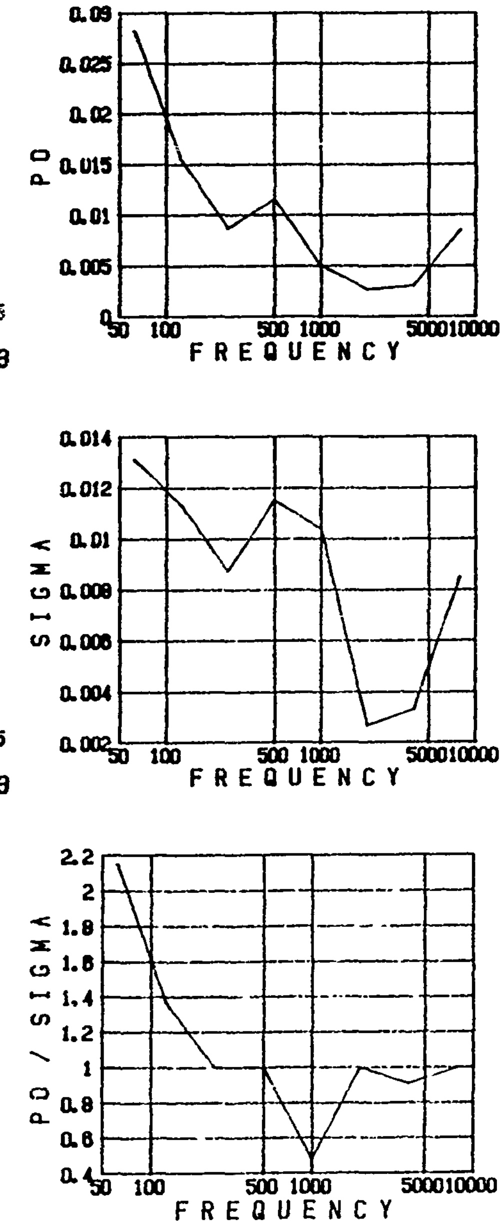


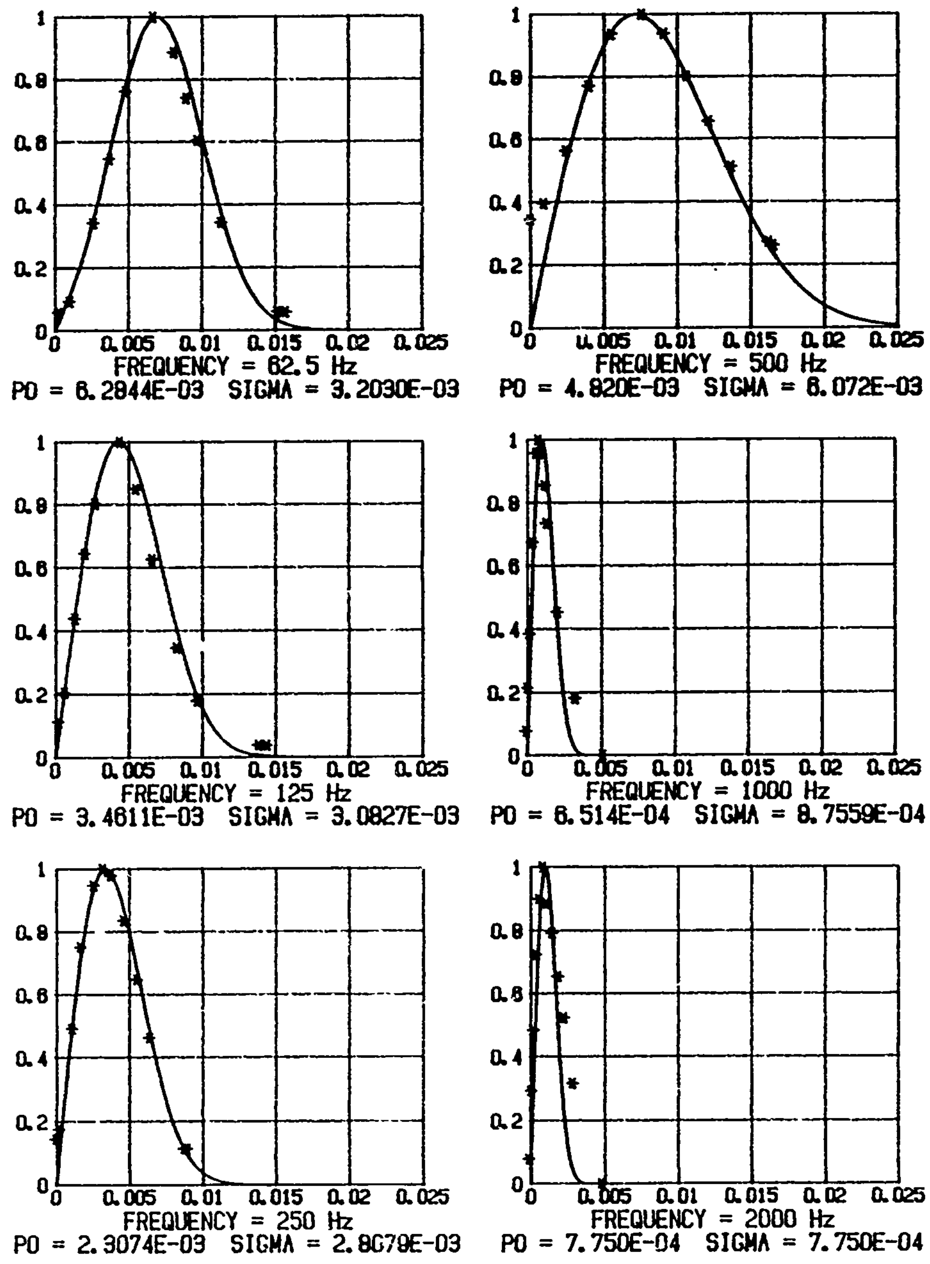
July 23, 1985
Run 1
Channel #1
Bondville, IL

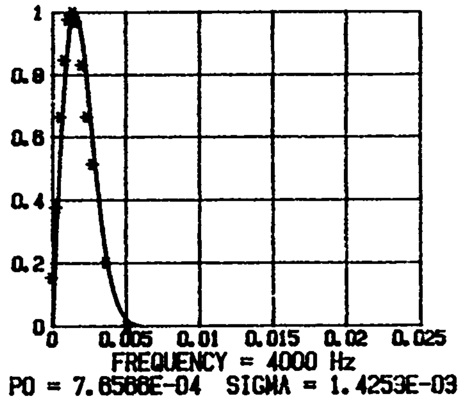


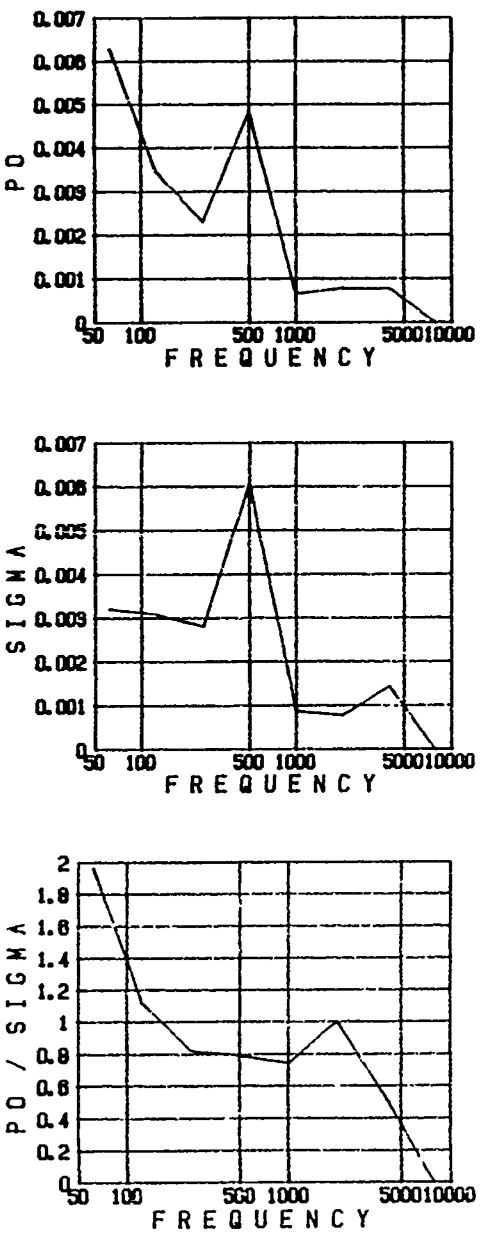


July 23, 1985
Run 1
Channel #3
Bondville, IL

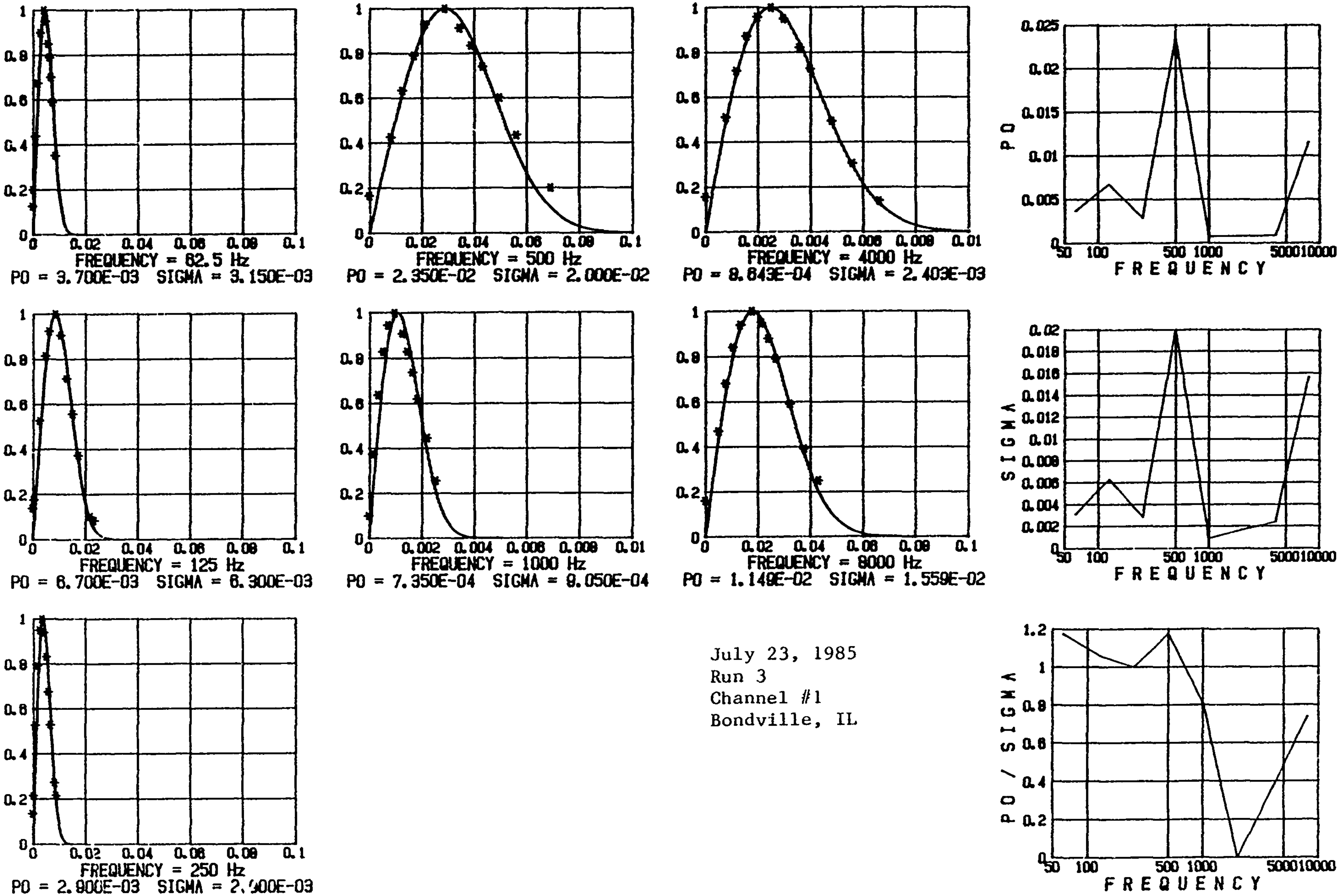




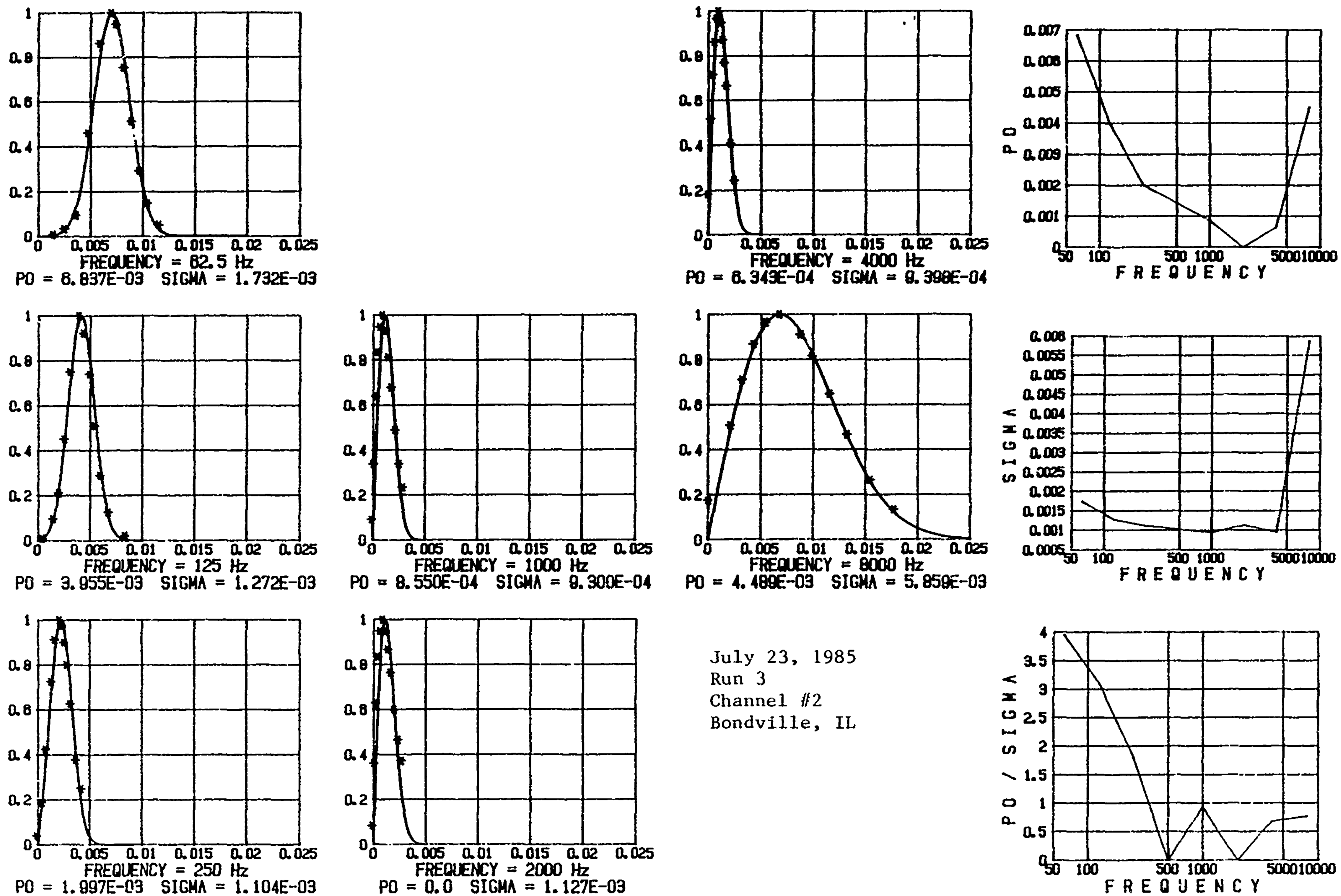




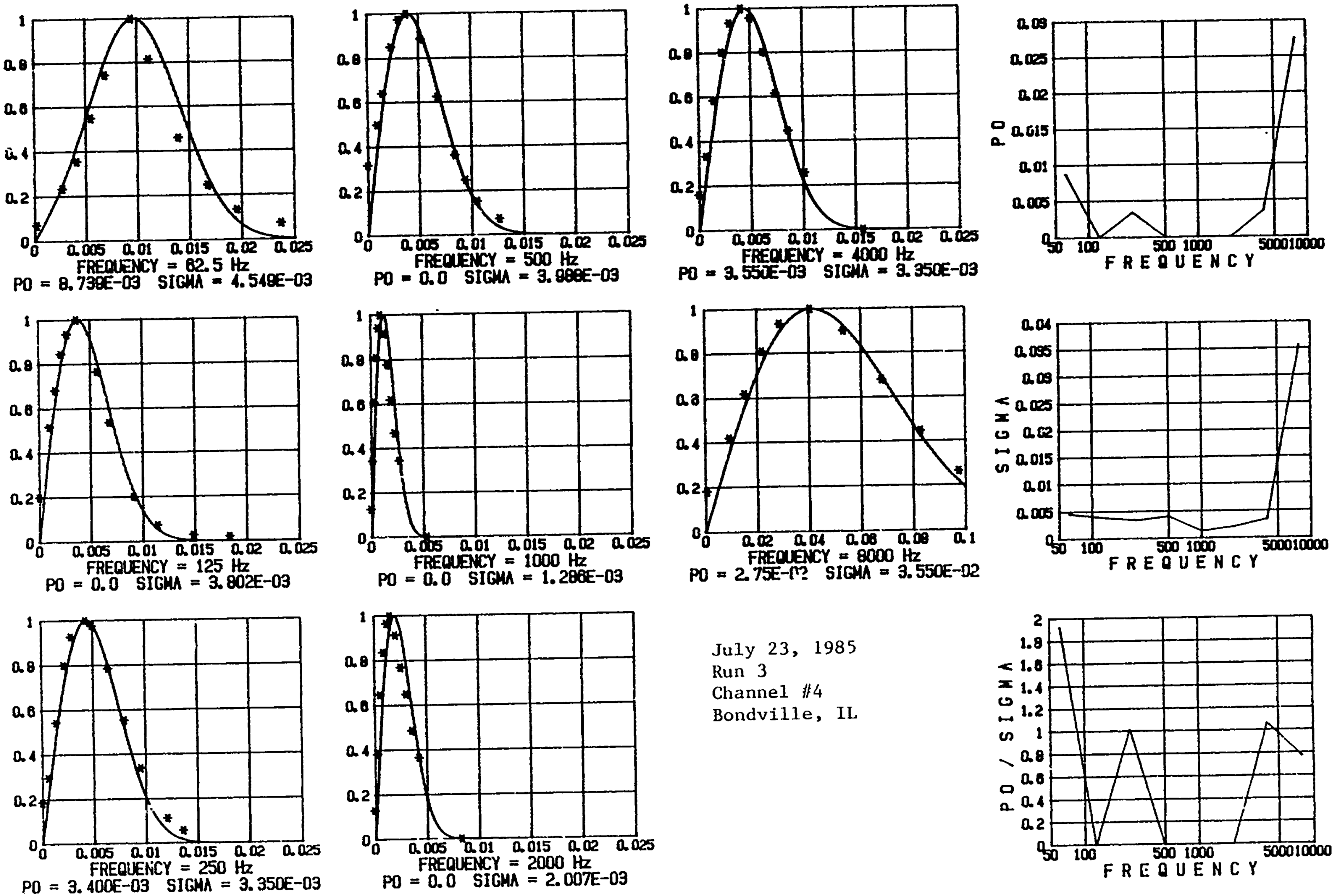
July 23, 1985
Run 2
Channel #1
Bondville, IL



July 23, 1985
Run 3
Channel #1
Bondville, IL



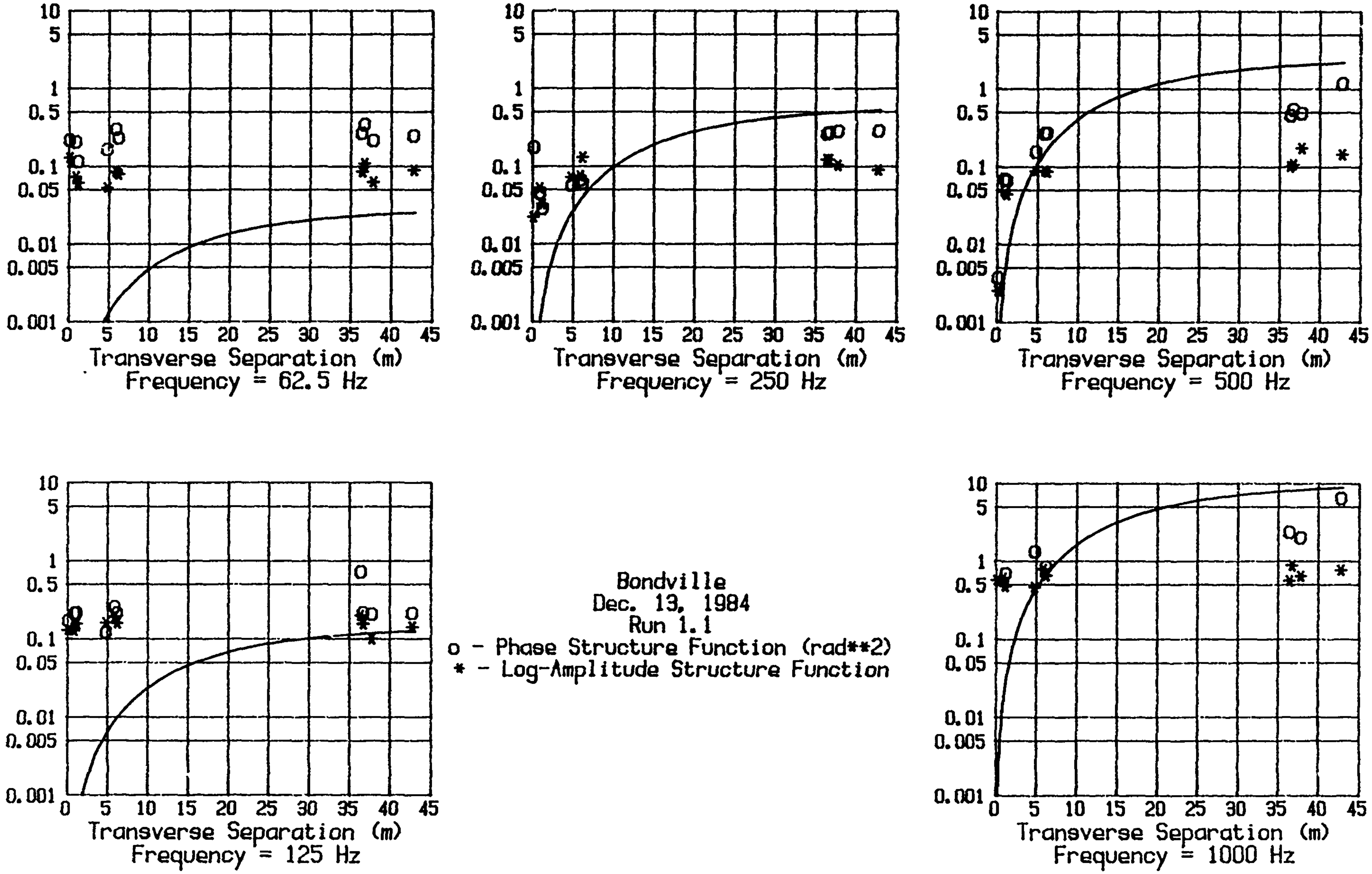
July 23, 1985
Run 3
Channel #2
Bondville, IL



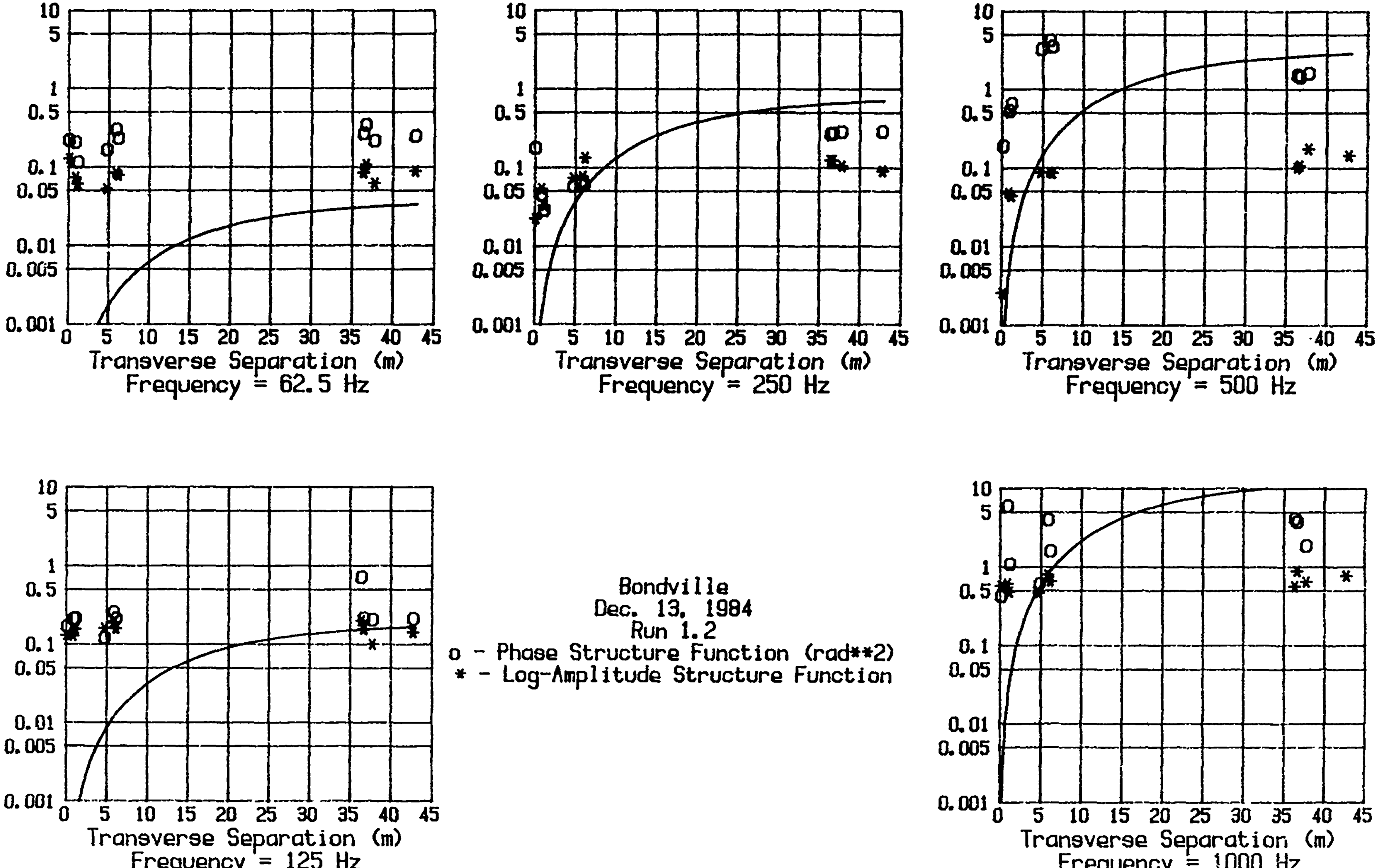
July 23, 1985
Run 3
Channel #4
Bondville, IL

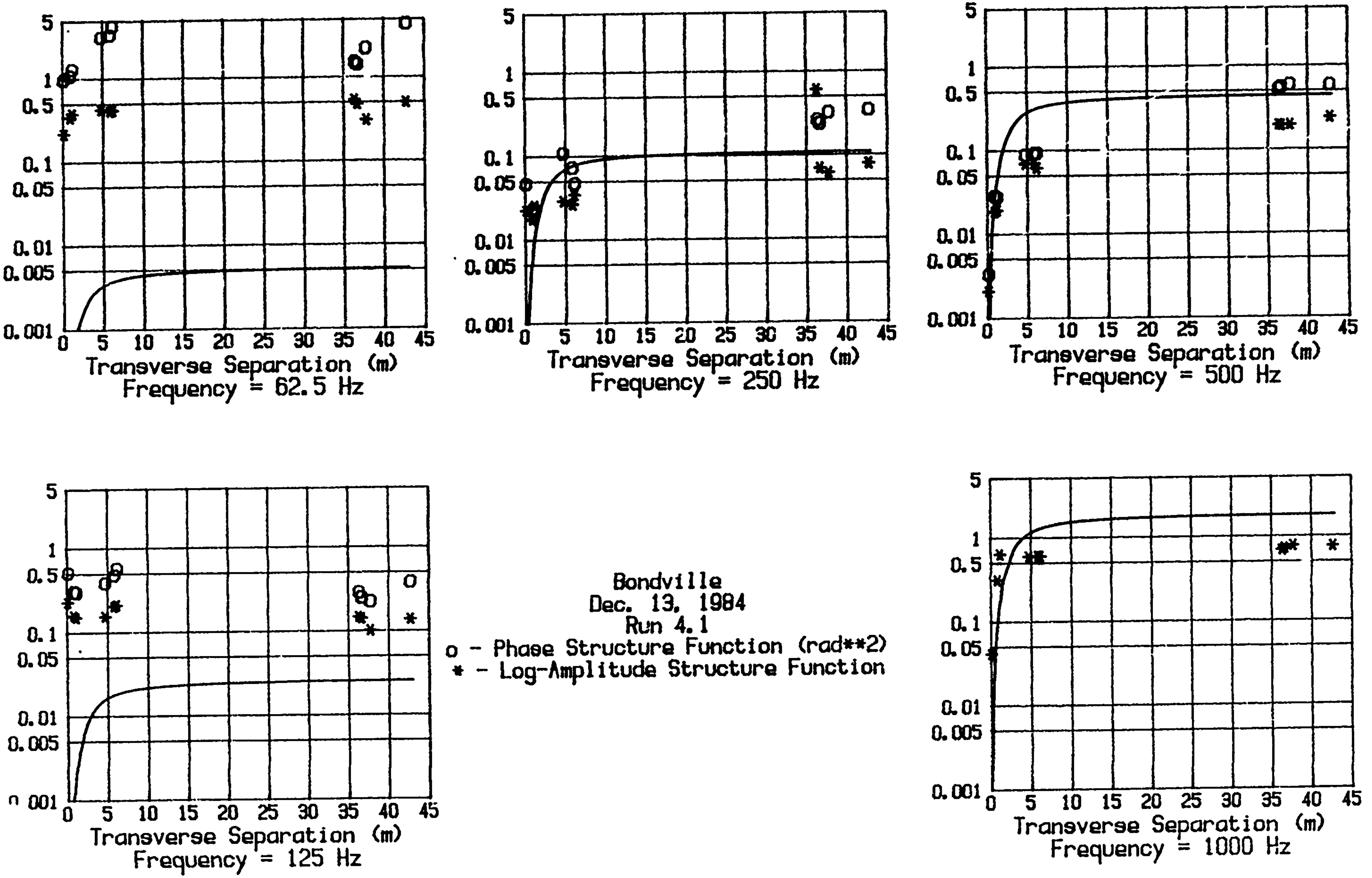
APPENDIX F

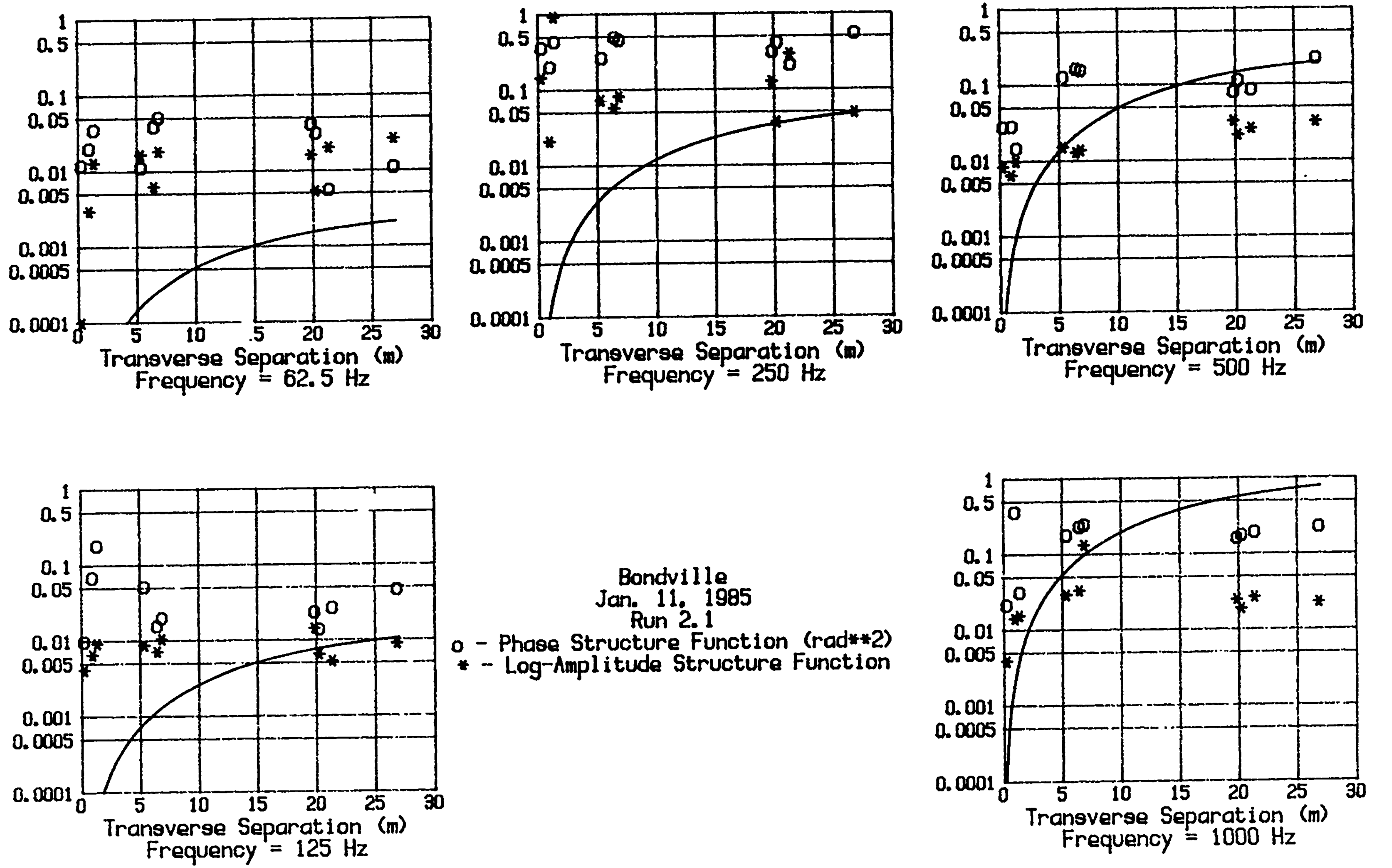
Comparisons between the structure functions calculated from the data and Daigle's theoretical structure function.

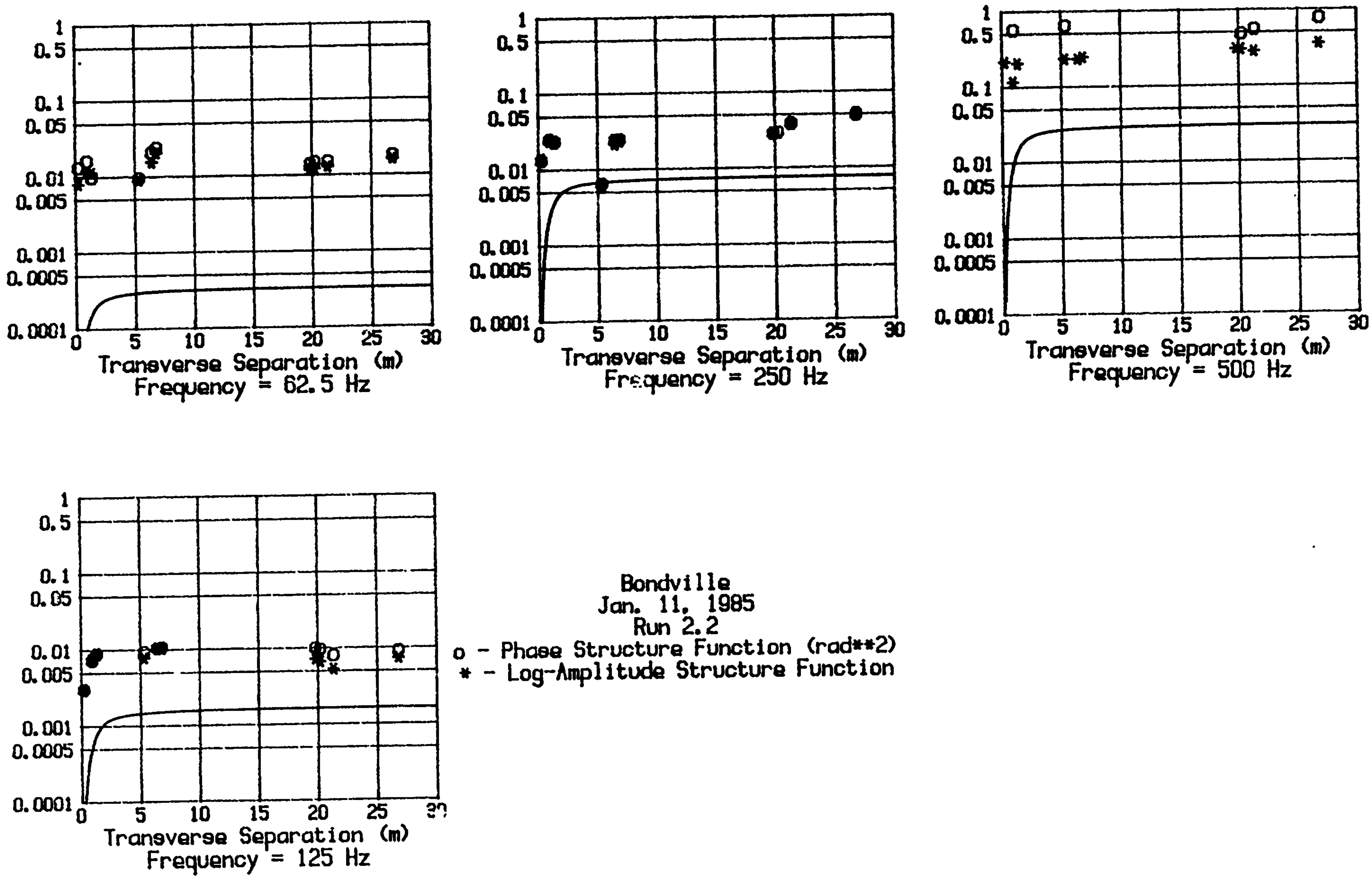


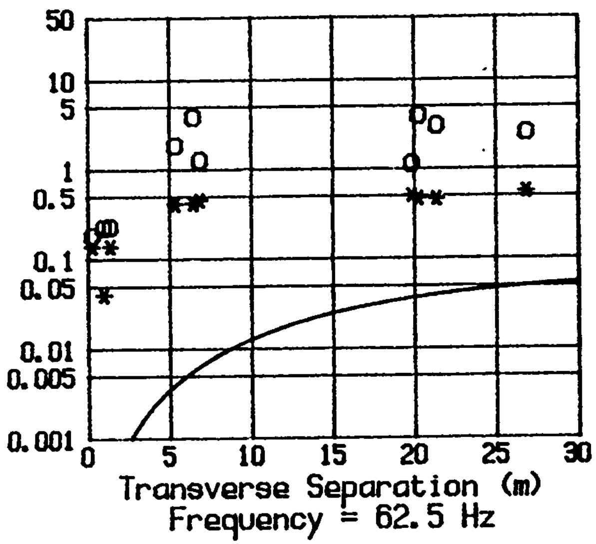
Transverse Separation (m)
Frequency = 62.5 Hz
Transverse Separation (m)
Frequency = 250 Hz
Transverse Separation (m)
Frequency = 500 Hz
Transverse Separation (m)
Frequency = 125 Hz
Bondville
Dec. 13, 1984
Run 1.1
o - Phase Structure Function (rad**2)
* - Log-Amplitude Structure Function
Transverse Separation (m)
Frequency = 1000 Hz

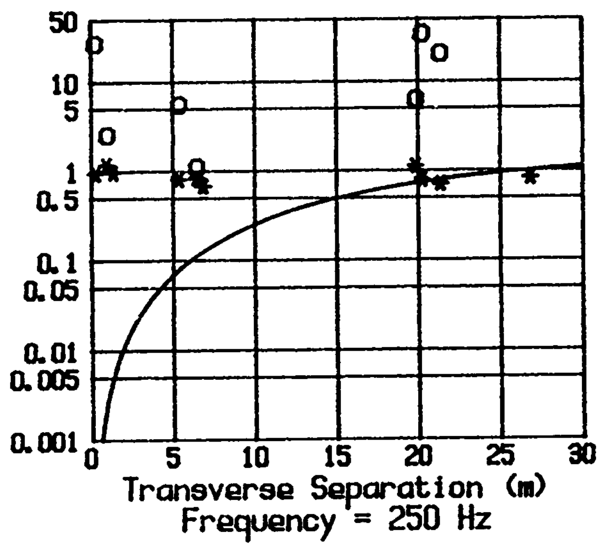

Transverse Separation (m)
Frequency = 62.5 Hz
Transverse Separation (m)
Frequency = 250 Hz
Transverse Separation (m)
Frequency = 500 Hz
Transverse Separation (m)
Frequency = 125 Hz
Transverse Separation (m)
Frequency = 1000 Hz
Bondville
Dec. 13, 1984
Run 1.2
o - Phase Structure Function (rad**2)
* - Log-Amplitude Structure Function

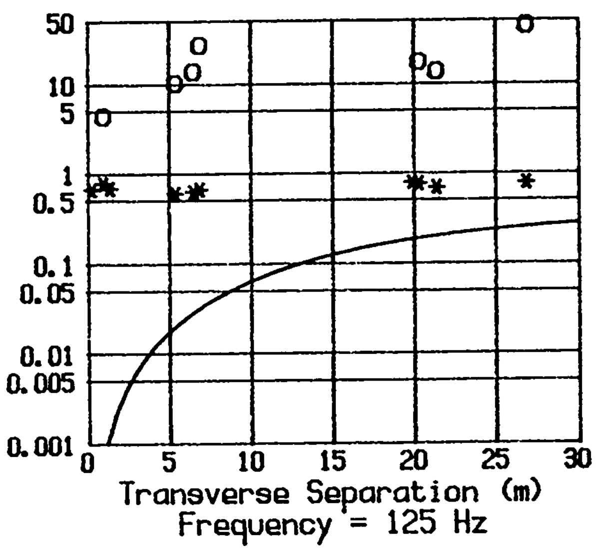

Transverse Separation (m)
Frequency = 62.5 Hz
Transverse Separation (m)
Frequency = 250 Hz
Transverse Separation (m)
Frequency = 500 Hz
Transverse Separation (m)
Frequency = 125 Hz
Transverse Separation (m)
Frequency = 1000 Hz
Bondville
Dec. 13, 1984
Run 4.1
o - Phase Structure Function (rad**2)
* - Log-Amplitude Structure Function


Transverse Separation (m)
Frequency = 62.5 Hz
Transverse Separation (m)
Frequency = 250 Hz
Transverse Separation (m)
Frequency = 500 Hz
Transverse Separation (m)
Frequency = 125 Hz
Transverse Separation (m)
Frequency = 1000 Hz
Bondville
Jan. 11, 1985
Run 2.1
o - Phase Structure Function (rad**2)
* - Log-Amplitude Structure Function

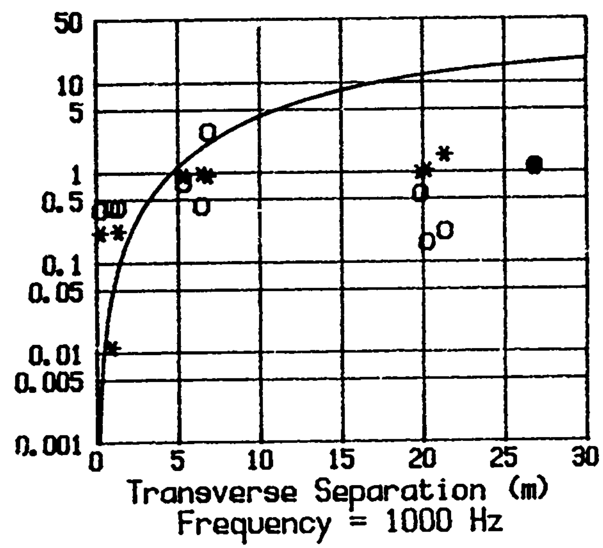

Transverse Separation (m)
Frequency = 62.5 Hz
Transverse Separation (m)
Frequency = 250 Hz
Transverse Separation (m)
Frequency = 500 Hz
Transverse Separation (m)
Frequency = 125 Hz
Bondville
Jan. 11, 1985
Run 2.2
o - Phase Structure Function (rad**2)
* - Log-Amplitude Structure Function

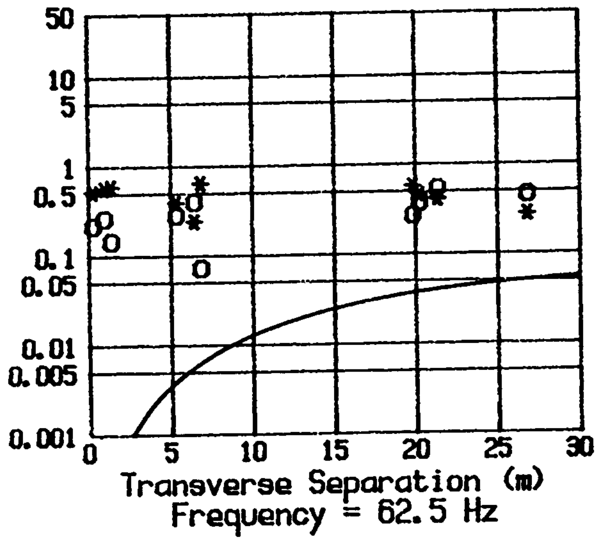


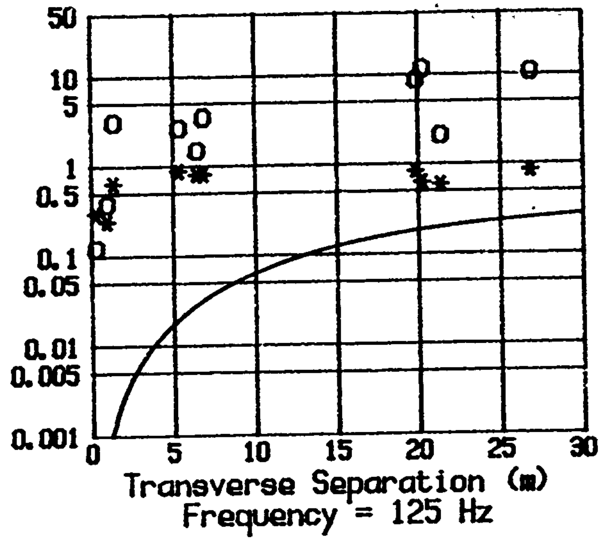


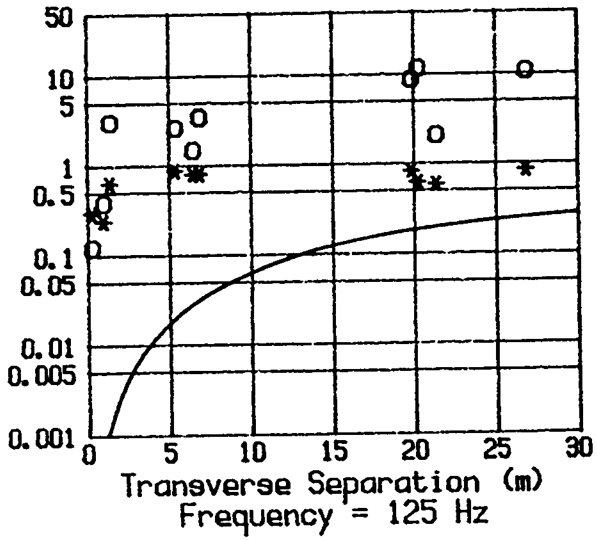


Bondville
July 23, 1985
Run 1
o - Phase Structure Function (rad**2)
* - Log-Amplitude Structure Function

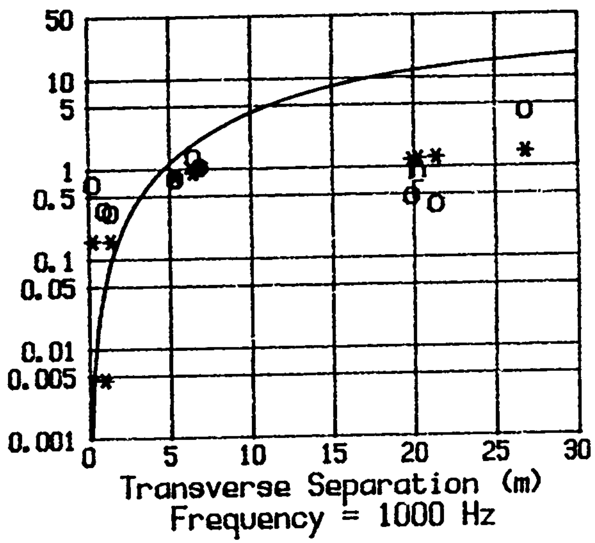


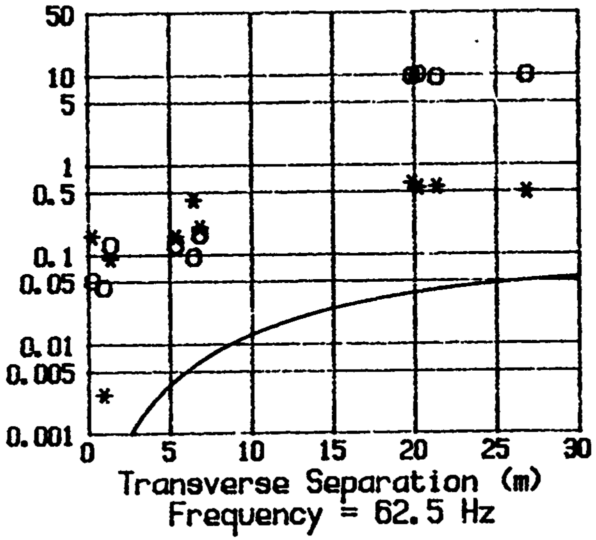


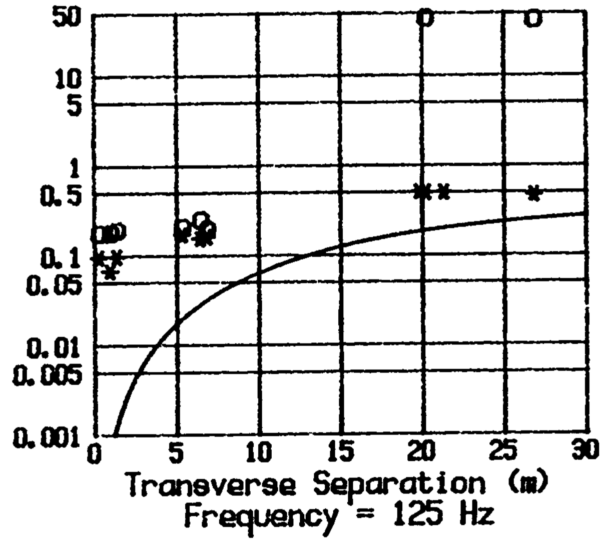


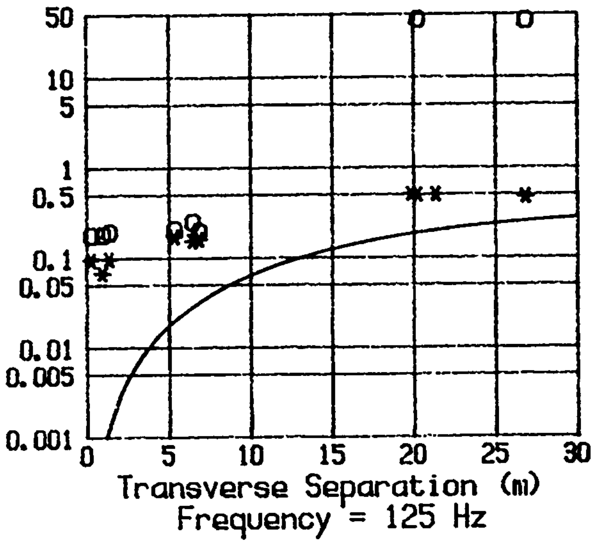


Bondville
July 23, 1985
Run 2
o - Phase Structure Function (rad**2)
* - Log-Amplitude Structure Function

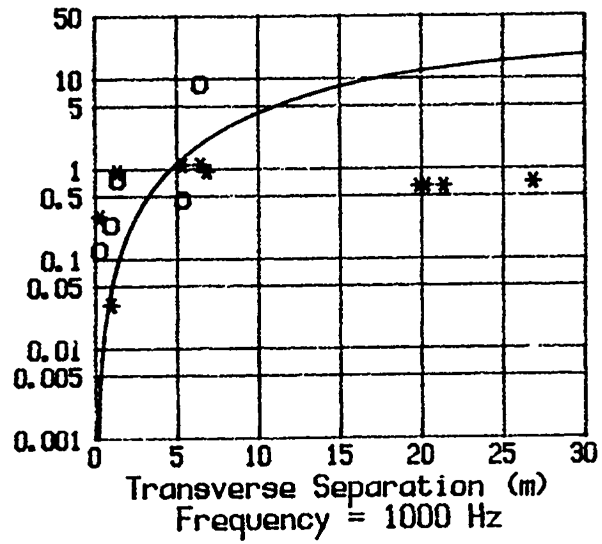


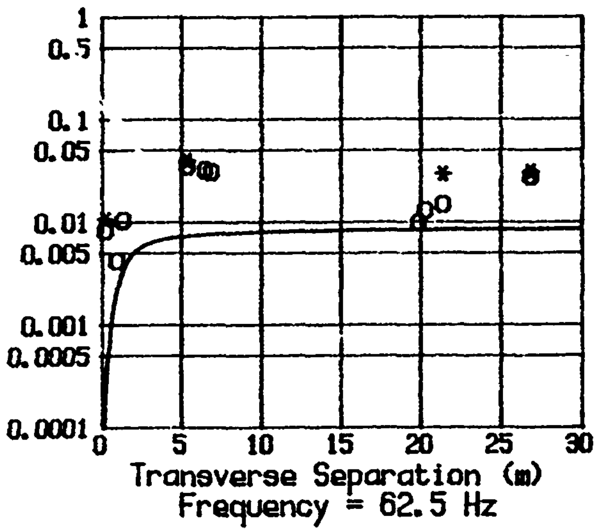


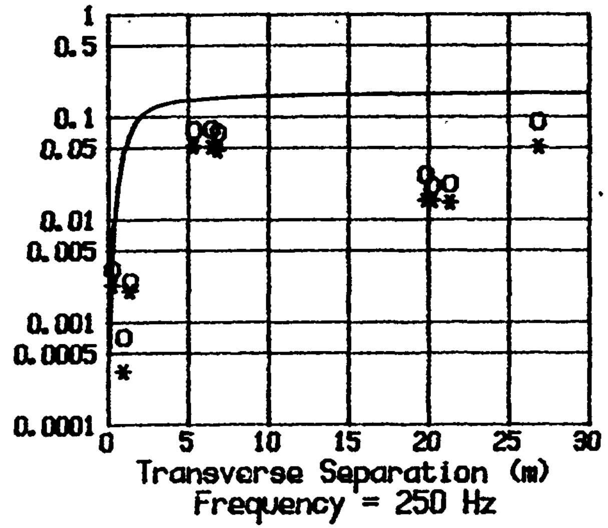


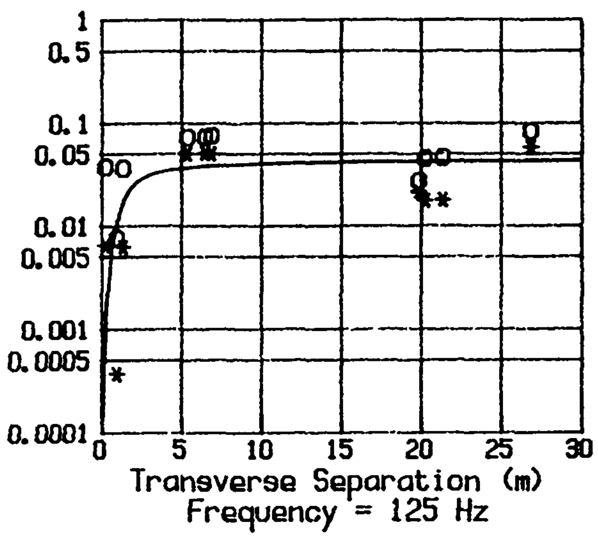


Bondville
July 23, 1985
Run 3
o - Phase Structure Function (rad**2)
* - Log-Amplitude Structure Function

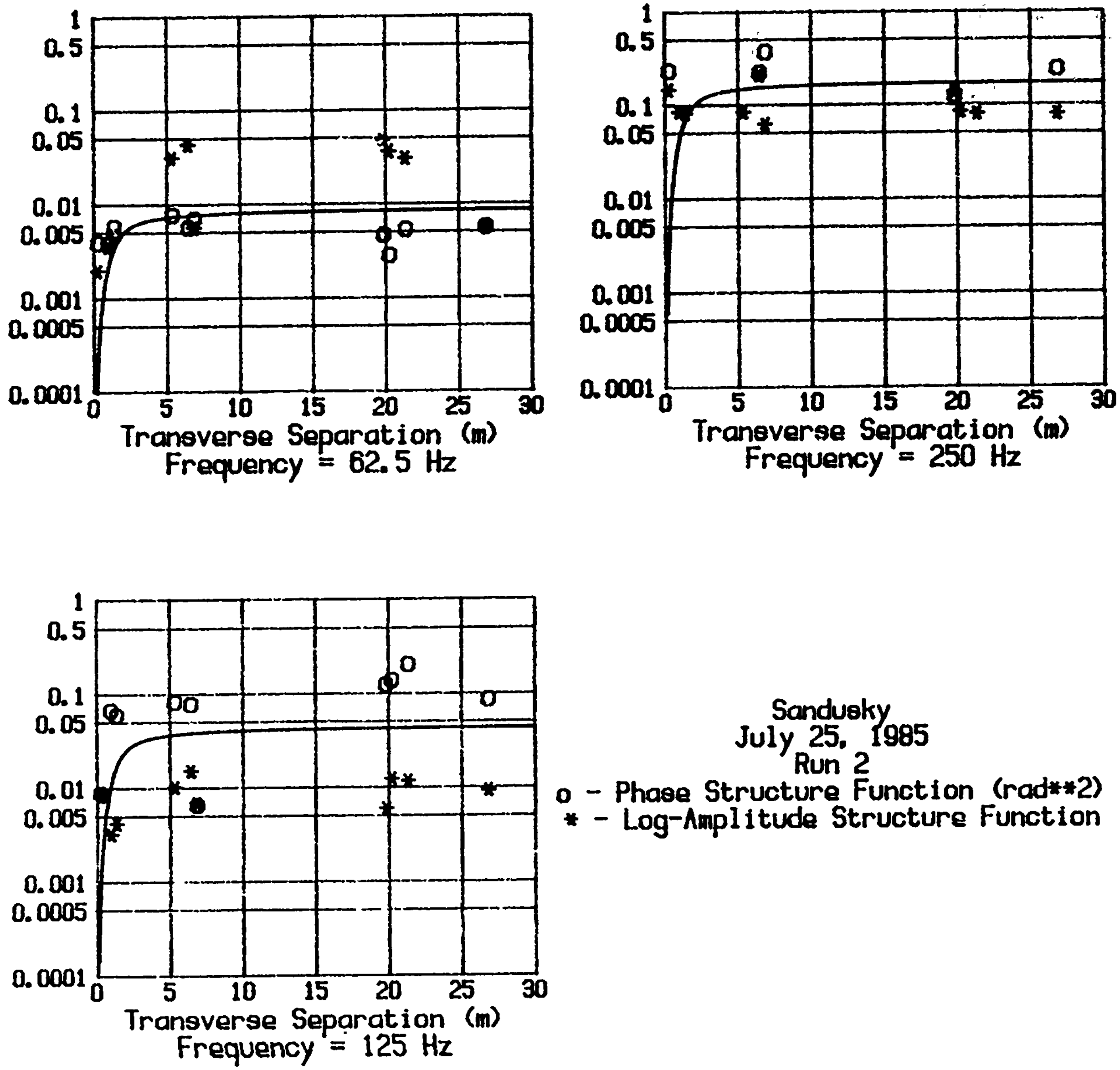


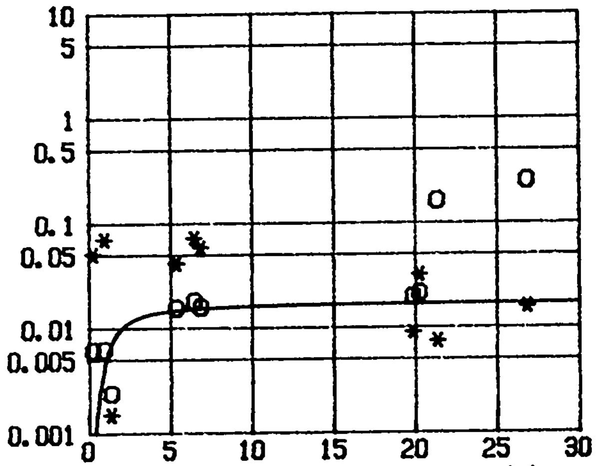




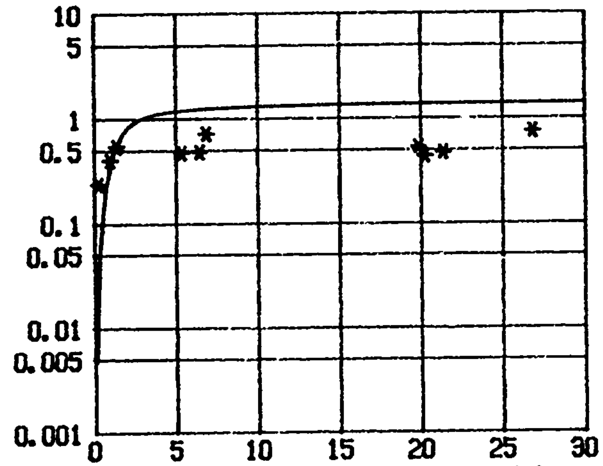


Sandusky
July 25, 1985
Run 1
o - Phase Structure Function (rad**2)
* - Log-Amplitude Structure Function

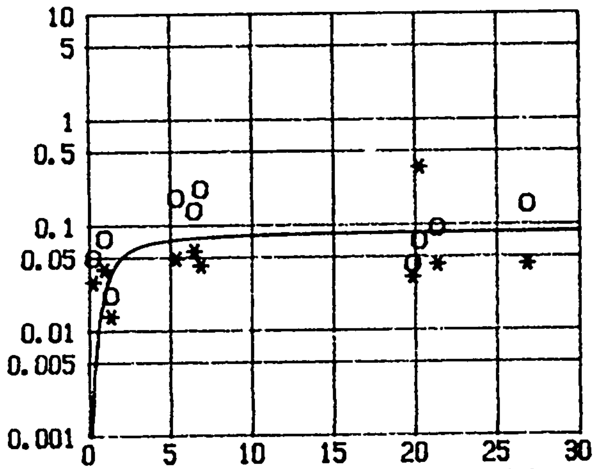

1
0.5
0.1
0.05
0.01
0.005
0.001
0.0005
0.0001
0
5
10
15
20
25
30
Transverse Separation (m)
Frequency = 62.5 Hz
Transverse Separation (m)
Frequency = 250 Hz
Transverse Separation (m)
Frequency = 125 Hz
Sandusky
July 25, 1985
Run 2
o - Phase Structure Function (rad**2)
* - Log-Amplitude Structure Function



Transverse Separation (m)
Frequency = 62.5 Hz

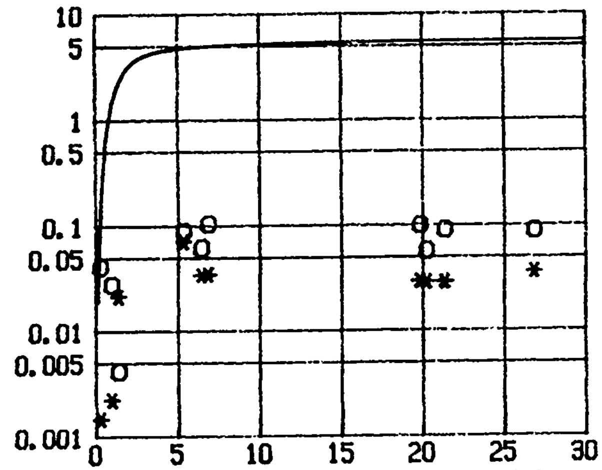


Transverse Separation (m)
Frequency = 500 Hz



Transverse Separation (m)
Frequency = 125 Hz

Sandusky
July 25, 1985
Run 3
o - Phase Structure Function (rad**2)
* - Log-Amplitude Structure Function



Transverse Separation (m)
Frequency = 1000 Hz